# DISTRIBUTED INTEGER PROGRAMMING

A Dissertation
Presented to
The Academic Faculty

By

Ezgi Karabulut

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

December 2017

**DISTRIBUTED INTEGER PROGRAMMING**

Approved by:

Dr. George Nemhauser, Advisor
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Dr. Shabbir Ahmed, Advisor
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Dr. Natashia Boland
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Dr. Santanu Dey
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Dr. Bistra Dilkina
School of Computational Sciences and Engineering
*Georgia Institute of Technology*

Date Approved: August 7, 2017

*To my late uncle.*

## ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisors Prof. George Nemhauser and Prof. Shabbir Ahmed for their constant support, guidance and motivation through these challenging times. I am extremely grateful for their mentorship during this the Ph.D. process, and hope to be worthy of them in the future.

I also thank my committee members Prof. Natashia Boland, Prof. Santanu Dey and Prof. Bistra Dilkina for their valuable comments and contributions to this thesis.

I have had so many great friends during these 5 years. Their company has been my primary support, some more than others. Unfortunately I will not be able to name them all here, but I know that we have a far stronger bond than these pages. As I promised, I also thank the Atlanta Symphony Orchestra for providing me with weekly inspiration on my research.

Last, but not the least, I am mostly grateful to my family. Georgia Tech has not only given me a future career, but also a future family, my soon-to-be husband Melih. Having him by my side made all the difference. Being away from home for five years has been tough, both on me and my family. I apologize for all the times that I couldn't be there for them. This thesis is on them, as much as it is on me.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## SUMMARY

In this thesis, we study distributed integer programming problems that involve multiple players with integer programming problems linked together with a common resource constraint. Our goal is to design decentralized algorithms that do not require a central processor to allocate the resource across the players to solve the overall problem. The algorithms that we design have optimality guarantees when applied to problems for which the marginal value of each additional resource is non-increasing. For problems that do not have this step-wise concave structure, we propose approximation algorithms and provide error bounds. We also perform experiments to evaluate the algorithms' average performance on problems without the desired structure. Finally, we consider the same problem in an online setting. We show that there exists no deterministic online algorithms for our problem that has the state of the art error bound. Therefore we propose a randomized decentralized online algorithm for our problem whose error bound matches the results in the literature.

# CHAPTER 1
# INTRODUCTION

## 1.1 Distributed Optimization

With the advent of modern applications involving large and complex data sets, there is increasing need for decentralized storage of data along with distributed solution methods for their processing. This need has given rise to the field of distributed optimization, where an optimization problem is solved with multiple processors (interchangeably referred to as *agents* or *players* throughout the thesis) communicating with each other without the need of a central coordinator. Research in this area focuses on how to distribute the data to increase algorithmic efficiency, how to select a smaller subset of the data to solve the problem with minimal loss of accuracy, or how to parallelize the algorithm in order to speed it up. Besides big data needs, distributed optimization may also arise in the setting of multiple players who are cooperating to solve a common problem but do not want to fully share their private data hence making it impossible to solve the problem with a centralized processor. The challenge is to solve these problems with high accuracy and limited communication when the data is distributed among multiple processors.

An important application area of distributed optimization is sensor networks. Sensor networks consists of sensors distributed over a topology, that collect data and communicate among each other. They are designed to collect large amounts of data, therefore it is neither feasible, nor efficient to communicate the data from every sensor to a central node and store it there. Hence, distributed optimization algorithms are devised to solve optimization problems that arise in the sensor networks. The most basic optimization problem in a sensor network is parameter estimation [1], such as the average of the data collected [2]. This problem can be generalized to problems such as robust estimation, i.e. detecting

outlier data and discarding it from the estimation process, or the problem of locating the source of an acoustic signal.

Various network optimization problems also have applications in a distributed setting [3]. For example, in network flow problem communication may be allowed only between neighboring nodes, and there exists no central node.

Another problem that is targeted in a distributed setting is regression anaylsis [4]. In such multi-agent systems, where every agent has access to a portion of the data, a distributed optimization algorithm is required to apply regression. Different algorithms are available when the data is distributed vertically, i.e. every agent only knows a subset of the components of all data vectors, or horizontally, i.e. every agent knows all of the components of a subset of the data vectors.

Other than these particular applications, there is a focus in the literature on the distributed versions of some general convex optimization algorithms. Subgradient and ADMM algorithms have received the most attention among the optimization methods. Studies by Nedic and Ozdaglar [5], Lobel et al. [6] and Raffard et al. [7] cover the distributed version of subgradient methods. Wei and Ozdaglar [8] and Boyd et al. [9] provide analysis for distributed ADMM algorithms. Distributed algorithms based on dual subgradient averaging [10], Newton-type distributed algorithms [11], and distributed gradient algorithms based on the Nesterov gradient [12] have also been studied. The important thing about these convex optimization algorithms is that they are generally designed to have coordinate-based updates at most of the steps, which is very convenient for decentralization.

These algorithms can be applied to solve any such optimization problem in the application areas discussed above. In addition to these application areas, they are also used for distributed computer networks when the problem data is too big and is distributed among multiple processors.

Much of the work on distributed optimization has focused on convex optimization. To the best of our knowledge there has been very limited progress in the discrete setting. Most

of the work in this field focuses on submodular set functions, i.e. functions $f$ defined on the ground set $N$ that satisfy $f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T)$ for all $S \subseteq T \subseteq N$ and $i \in N \backslash T$. The main motivation behind the distributed setting is the large amounts of data stored in multiple processors, and instead of solving the optimization problem over the whole data set, a representative subset of the data is selected to solve the main problem. Mirzasoleiman et al. [13] study a greedy approach to solve the problem of maximizing a submodular set function with respect to a cardinality constraint, i.e. $|S| = k$. Barbosa et al. [14] enhance this approach by randomizing the process of data distribution to the processors; and generalizing the cardinality constraint to matroid partition, i.e. $\cup_i S_i = N$ and $S_i \cap S_j = \emptyset$, knapsack, i.e. $\sum_{i \in S} a_i \leq b$, and $p$-system constraints, i.e. $S \in \cap_{i=1}^{p} \mathcal{F}_p$, where $\mathcal{M}_i = (N, \mathcal{F}_i)$ are matroids for $i = 1, \ldots, p$. Singh and O'Keefe [15] consider a scheduling problem in a decentralized supply chain, and underline the importance of different agents coordinating among themselves.

### 1.2 Online Optimization

The last chapter of this thesis focuses on online optimization. Many of the supervised machine learning problems in online settings can be formulated as online optimization problems, where the goal is to find the model that best fits the data when the training data points are encountered one at a time. The critical distinction between the online and the classical offline problems is that in the online setting, the decision is fixed prior to observing all of the data. Applying regression to separate spam e-mails from non-spam emails, portfolio maximization problems where the goal is distributing the assets over a set of investments to maximize expected gain, estimation of the missing elements in a preference matrix, i.e. knowing user $i$ likes song $j$ but has no input about song $k$, how can you estimate their preference for song $k$, are some popular application areas for online optimization [16].

Some of these application areas can be encountered in distributed settings, e.g. consider

the portfolio maximization problem where the investments correspond to different agents, or the preference estimation problem where the data is distributed over multiple processors. Many digital resource sharing problems that are solved repeatedly over time, such as allocating advertisement slots on a web search page where the advertisements belong to different players, qualify as decentralized online optimization problems considered in this thesis. The main challenge in online optimization is making the decision prior to observing the problem data. Therefore, instead of finding the optimal resource allocation, the goal becomes finding a series of resource allocations that have an upper bound on the cumulative distance from the best fixed resource allocation.

There is a vast literature on online convex optimization algorithms, explained thoroughly in [16] and [17]. They range between relatively simple first order optimization algorithms, such as gradient descent, to more complicated regularized methods, where the straightforward follow-the-leader approach, i.e. choose the optimal decision with respect to the previous data that is already observed, is modified. There are also variations of the algorithms depending on the feedback type, i.e. whether the function itself or the function value of the decision made is observed at the end of each iteration. These algorithms assume a convex feasible solution set, and propose a smooth transition over the variables throughout the iterations. Unfortunately, our problem requires discrete resource allocations at every iteration, making these online convex optimization algorithms not directly applicable.

Relatively less work has been done on online discrete optimization. Koolen et al. [18] and Audibert et al. [19] focus on the problem of minimizing a linear function over a subset of the binary hypercube in an online setting. They don't assume anything on the structure of the binary hypercube, i.e. the constraints, therefore their algorithms are open for specialization knowing the feasible solution set structure. The area of online optimization includes deterministic algorithms as well as randomized algorithms.

## 1.3 Contribution of thesis

In this thesis we consider multi-player discrete optimization problems where the players share a single resource. We design decentralized algorithms that do not require a central processor to allocate the resource across the players to solve the overall problem. The algorithms that we design have optimality guarantee for problems with certain structures. We further design approximation algorithms for problems that do not have the desired structure. Our approach in designing the approximation algorithms can be divided into two branches: using approximation functions throughout the original decentralized algorithms, and modifying the decentralized algorithms to accommodate the structure of the problem.

In the context of online optimization, we consider both deterministic and randomized algorithms as candidates while proposing our decentralized online optimization algorithm. The performance of our algorithm matches the state of the art error bound. We also analyze the performance of approximation algorithms under an online setting.

This thesis is structured in the following way: In Chapter 2, we propose decentralized algorithms for the distributed resource sharing problems, and provide sufficient conditions on the problems for the algorithm to yield an optimal solution. Chapter 3 includes approximation algorithms for problems that do not satisfy the sufficient condition, and their performance guarantees. Chapter 4 focuses on online optimization, namely deterministic and randomized algorithms for an online resource sharing problem. Chapter 5 gives conclusions.

# CHAPTER 2

# OPTIMAL ALGORITHMS FOR DISTRIBUTED INTEGER PROGRAMMING PROBLEMS

## 2.1 Introduction

We consider a separable integer program with a single coupling cardinality constraint in the form of (2.1)

$$z = \max \left\{ \sum_{i=1}^{m} f_i(x_i) : \sum_{i=1}^{m} a_i^T x_i \leq K \ , \ x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \quad i = 1, \ldots, m \right\}. \quad (2.1)$$

This problem can be interpreted as a collection of independent player optimization problems coupled by a resource constraint. Independent players sharing a common resourse is a prevalent phenomenon. Many budgeting problems involving multiple players are of this form. Some applications of this problem can be found in areas such as network optimization and portfolio optimization. Without loss of generality, we assume all data is integer. Here, $i = \{1, \ldots, m\}$ is the set of players.

Given an allocation of the resource to the players, i.e. a collection $\{K_i\}_{i=1}^m$ such that $\sum_{i=1}^m K_i = K$, (2.1) can be partitioned into subproblems:

$$z_i(K_i) = \max \left\{ f_i(x_i) : a_i^T x_i \leq K_i \ , \ x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \quad (2.2)$$

Clearly, any arbitrary collection $\{K_i\}_{i=1}^m$ with $\sum_{i=1}^m K_i = K$, satisfies $\sum_{i=1}^m z_i(K_i) \leq z$. Furthermore, there exists an optimal allocation $\{K_i^*\}_{i=1}^m$ that satisfies $\sum_{i=1}^m K_i^* = K$ for which $\sum_{i=1}^m z_i(K_i^*) = z$. This means that when we have an optimal allocation $\{K_i^*\}_{i=1}^m$ of the resource, each player can solve their subproblem (2.2) independently, and find an

optimal solution of (2.1). In this chapter, we propose decentralized algorithms that exploit this property, and seek an optimal allocation of $K$ to solve (2.1) in a distributed manner with limited information sharing. We derive specialized algorithms designed for the special cases where the coupling constraint is a cardinality constraint, or where the objective function is in the form of cardinality.

## 2.2 Literature Survey

Distributed algorithms have been widely used in convex optimization. Studies by Nedic and Ozdaglar [5], Lobel et al. [6] and Raffard et al. [7] focus on the distributed version of the subgradient method. Duchi et al. [10] propose distributed algorithms based on dual subgradient averaging. Wei and Ozdaglar [8] and Boyd et al. [9] study distributed ADMM algorithms. Jadbabaie et al. [11] work on an alternative approach to Newton-type distributed algorithms. Recent research by Jakovetic et al. [12] includes a distributed gradient algorithm based on the Nesterov gradient.

Much less research has been done on distributed optimization of discrete problems. Mirzasoleiman et al. [13] study a greedy approach to solve the problem of maximizing a submodular function subject to a cardinality constraint introduced in [20] by Nemhauser et al.. They propose a distributed framework by assuming the huge data at hand is distributed among the processors, and every processor must first choose a set of candidate solutions. The central processor then applies the greedy algorithm on this restricted set, instead of the complete data set. Barbosa et al. [14] enhance this approach by randomizing the process of data distribution to the processors; and generalizing the cardinality constraint in [20] to matroid, knapsack, and p-system constraints. Singh and O'Keefe [15] work on a different area in distributed discrete optimization. They consider a scheduling problem, and solve the Lagrangean relaxation of that problem in a decentralized way.

Bertsekas [21] provides a comprehensive tutorial for auction algorithms for network problems. Although these auction algorithms are not necessarily restricted to be distribu-

ted; most of his algorithms are easily distributable. His idea of an auction is similar to the decentralized algorithms that we propose; however our underlying problem structure differs greatly from his. He introduces an auction algorithm for the assignment problem and shows the reductions of the shortest path and transportation problems to the assignment problem, and the modifications on the auction algorithm for these special cases.

We are interested in optimizing discrete problems that use a common resource, namely integer programming problems coupled with a knapsack constraint. In Bertsekas' auction algorithm setting, the players all have a common contraint set; their decisions are highly dependent on each other's, which is not the case in our setting. In our problems, once the resource is distributed among the players, their corresponding problems become completely independent.

### 2.3 Distributed Problems with a Cardinality Constraint

We are given problem (2.3),

$$z = \max \left\{ \sum_{i=1}^{m} w_i^T x_i : \sum_{i=1}^{m} \mathbf{1}^T x_i \leq K,\ x_i \in \{0,1\}^{N_i}\ \ i = 1, \ldots, m \right\} \tag{2.3}$$

which is a cardinality knapsack problem in which the items are distributed among $m$ players, and each player is assigned $N_i$ items that they can choose from. The decision variable representing whether or not player $i$ chooses their $j^{\text{th}}$ item is $x_{ij}$, and the value of that decision is $w_{ij}$. The players are expected to coordinate with each other in order to select $K$ items in total.

Once the resource level $K$ is distributed among the players, such that player $i$ gets $K_i$ units of resource, each player gets their individual problem (2.4):

$$z_i(K_i) = \max \left\{ w_i^T x_i : \mathbf{1}^T x_i \leq K_i\ ,\ x_i \in \{0,1\}^{N_i} \right\}. \tag{2.4}$$

Without loss of generality, we assume the indices of $x_i$ are in decreasing order of their

objective function coefficients; i.e. $w_{ij} \geq w_{i,j+1}$ for $j = 1 \dots N_i - 1$. Thus the optimal solution $x_i^*$ of (2.4) has $x_{ij}^* = 1$ for $j = 1, \dots, K_i$ and $x_{ij}^* = 0$ for $j = K_i + 1, \dots, N_i$.

### 2.3.1 The CC-Lin Algorithm

The first decentralized algorithm we propose for this setting works as follows: Each player is given an initial resource level $K_i$ that satisfies $\sum_{i=1}^{m} K_i = K$. They share with each other how much their objective will increase if their resource is increased by one unit, $\Delta_i^+$, which corresponds to $w_{i,K_i+1}$, and how much their objective will decrease if their resource is decreased by one unit, $\Delta_i^-$, which corresponds to $w_{i,K_i}$. In order to ensure that the highest increase and lowest decrease are unique, players add a small value $\epsilon(i)$ to their $\Delta_i^+$ and $\Delta_i^-$ values as a tie-breaker. The player with the highest increase ($i^+$) and the player with the smallest decrease ($i^-$) swap a single unit of resource, i.e. $K_{i^+}$ becomes $K_{i^+} + 1$ and $K_{i^-}$ becomes $K_{i^-} - 1$. In the subsequent convergence analysis, we prove that the players $i^+$ and $i^-$ are distinct unless it is the last iteration. After each iteration, all players except for $i^+$ and $i^-$ remain at the same resource level, and only $i^+$ and $i^-$ with the new resource levels need to update and share their $\Delta_i^+$ and $\Delta_i^-$ values. The swapping of the resources continue until the highest increase in the objective value is not larger than the smallest decrease, and at that point the algorithm terminates. We refer to this decentralized algorithm as CC-Lin, short for *cardinality constraint linear convergence*, and describe it in Figure 2.1.

*Convergence and Optimality*

In Line 4, as a tie-breaking rule the players add a function of their indices, $\epsilon(i)$, to the $\Delta_i^+$ and $\Delta_i^-$ values. An example for this function might be $\epsilon(i) = \varepsilon i$ for some $\varepsilon \approx 0$. We assume that $\epsilon(i)$ is significantly smaller in magnitude compared to $\Delta_i^+$ and $\Delta_i^-$; therefore it will not be included in the convergence and optimality analyses for the sake of simplicity.

Our main result on the performance of the CC-Lin algorithm is stated in Theorem 2.1.

**Input:** Player index $i \in \{1 \dots m\}$, initial resource allocation $K_i$
**Output:** Resource allocation $K_i$

**while true do**
  **if** isChanged== **true then**
    isChanged= **false**
    $\Delta_i^+ = z_i(K_i + 1) - z_i(K_i) + \epsilon(i)$; $\Delta_i^- = z_i(K_i) - z_i(K_i - 1) + \epsilon(i)$
    BROADCAST$(\Delta_i^+, \Delta_i^-)$
  **end if**
  RECEIVE$\Big(\left\{\Delta_p^+, \Delta_p^-\right\}_{p \subseteq \{1,\dots,m\}}\Big)$
  $\Delta^+ = \max_{p \in \{1 \dots m\}} \left\{\Delta_p^+\right\}$; $i^+ = arg\max_{p \in \{1 \dots m\}} \left\{\Delta_p^+\right\}$
  $\Delta^- = \min_{p \in \{1 \dots m\}} \left\{\Delta_p^-\right\}$; $i^- = arg\min_{p \in \{1 \dots m\}} \left\{\Delta_p^-\right\}$
  **if** $\Delta^- \geq \Delta^+$ **then**
    **return** $K_i$
  **end if**
  **if** $i^+ == i$ **then**
    $K_i = K_i + 1$; isChanged= **true**
  **else if** $i^- == i$ **then**
    $K_i = K_i - 1$; isChanged = **true**
  **end if**
**end while**

Figure 2.1: The CC-Lin algorithm

**Theorem 2.1.** *When the CC-Lin algorithm is applied to problem (2.3), it outputs an optimal resource allocation in at most $K$ iterations.*

The following results are used to build up to the proof of Theorem 2.1.

**Lemma 2.1.** When the CC-Lin algorithm is applied to problem (2.3), the players $i^+$ and $i^-$ are distinct unless it is the last iteration.

*Proof.* Assume $i = i^+ = i^-$, with resource value $K_i$. $\Delta_i^+ = w_{i,K_i+1} = \Delta^+$ and $\Delta_i^- = w_{i,K_i} = \Delta^-$ hold by our assumption. As $w_{i,K_i+1} \leq w_{i,K_i}$, we have $\Delta^+ \leq \Delta^-$, which implies that the algorithm terminates at this iteration. □

For the following proofs, let the superscript $(t)$ refer to the value of a parameter at iteration $t$ of the algorithm.

**Lemma 2.2.** When the CC-Lin algorithm is applied to problem (2.3), the sequence of $\Delta^+$ values throughout the iterations are nonincreasing; and the sequence of $\Delta^-$ values are nondecreasing.

*Proof.* We will prove the first part of Lemma 2.2 by showing that $\Delta^{+(t)} \geq \Delta^{+(t+1)}$ for all $t$ values. The second part, $\Delta^{-(t)} \leq \Delta^{-(t+1)}$ follows from a similar argument. Note that

$$\Delta^{+(t)} = \max_{i=1,\dots,m} \left\{ \Delta_i^{+(t)} \right\} = \Delta_{i^{+(t)}}^{+(t)}$$

holds by the definition of $\Delta^+$ and $i^+$, and this value equals $w_{i^{+(t)},K_{i^{+(t)}}+1}$, the value of the next best item player $i^{+(t)}$ will pick, with respect to their current resource level $K_{i^{+(t)}}$. We can write a similar definition of $\Delta^{+(t+1)}$ and expand it as:

$$\Delta^{+(t+1)} = \max_{i=1,\dots,m} \left\{ \Delta_i^{+(t+1)} \right\} = \max \left\{ \max_{i \neq i^{+(t)}, i^{-(t)}} \left\{ \Delta_i^{+(t)} \right\}, \Delta_{i^{-(t)}}^{-(t)}, \Delta_{i^{+(t)}}^{+(t+1)} \right\}.$$

First notice that, after iteration $t$, only the players $i^{+(t)}$ and $i^{-(t)}$ update their $\Delta_i$ values; therefore, except for those two players we have $\Delta_i^{+(t+1)} = \Delta_i^{+(t)}$. By the definition of $\Delta^+$ and $i^+$, $\max_{i \neq i^{+(t)}, i^{-(t)}} \left\{ \Delta_i^{+(t)} \right\} \leq \Delta^{+(t)}$. The optimality conditions of player $i^{-(t)}$ imply that the value of the item that they dropped at iteration $t$ must be equal to the value of the next best item to be selected at iteration $t+1$; hence $\Delta_{i^{-(t)}}^{+(t+1)} = \Delta_{i^{-(t)}}^{-(t)}$. $\Delta_{i^{-(t)}}^{-(t)} = \Delta^{-(t)} \leq \Delta^{+(t)}$ trivially holds, otherwise the algorithm would terminate at iteration $t$. Finally $\Delta_{i^{+(t)}}^{+(t+1)}$ equals the value of the $K_{i^{+(t)}} + 2^{\text{nd}}$ best item for player $i^{+(t)}$, $w_{i^{+(t)},K_{i^{+(t)}}+2}$; which is at most $w_{i^{+(t)},K_{i^{+(t)}}+1}$, or equivalently $\Delta^{+(t)}$. Thus $\Delta^{+(t+1)} \leq \Delta^{+(t)}$.

□

**Lemma 2.3.** When the CC-Lin algorithm is applied to problem (2.3), the changes in $K_i$ values are monotonous for every player $i$ throughout the algorithm.

*Proof.* Similar to the previous proof, we will only provide the proof for one side of the argument, namely that a player who increases their resource at some iteration cannot decrease

it at a later iteration. The other direction, i.e. a player who decreases their resource at some iteration cannot increase it at a later iteration, can be proven using a similar argument.

For contradiction, let player $i'$ be a player who has an increase of 1 unit in resource $K_{i'}$ at iteration $t'$, and a later decrease of 1 unit at iteration $t''$, and $K_{i'}$ stays constant in between iterations $t'$ and $t''$. We have shown in Lemma 2.2 that $\Delta^{+(t')} \geq \Delta^{+(t'')}$ as $t' \leq t''$. We define $t'$ to be the iteration when $i'$ is the player with the highest gain and $t''$ when $i'$ is the player with the smallest loss, therefore we know that $\Delta^{+(t')} = \Delta_{i'}^{+(t')}$ and $\Delta^{-(t'')} = \Delta_{i'}^{-(t'')}$. Furthermore, the value of the next best item picked at iteration $t'$ equals the value of the first item to be dropped until iteration $t''$, so $\Delta_{i'}^{+(t')} = \Delta_{i'}^{-(t)}$ for $t = t' + 1, \ldots, t''$. Using these arguments, we get the chain

$$\Delta^{-(t'')} = \Delta_{i'}^{-(t'')} = \Delta_{i'}^{+(t')} = \Delta^{+(t')} \geq \Delta^{+(t'')}. \tag{2.5}$$

As by our assumption, the algorithm has not terminated at iteration $t''$; which is only possible if $\Delta^{-(t'')} < \Delta^{+(t'')}$. Therefore (2.5) yields a contradiction, so there exists no such player $i'$.

□

*Proof of Theorem 2.1.* Let $\overline{K}$ be the resource allocation the CC-Lin algorithm outputs, and $K^*$ be an optimal resource allocation with minimum Hamming distance to $\overline{K}$. As a contradiction, assume that $\overline{K} \neq K^*$; and $\sum_{i=1}^{m} z_i(K_i^*) > \sum_{i=1}^{m} z_i(\overline{K}_i)$. Denote by $S^+$ the set of players $i$ with $K_i^* > \overline{K}_i$, and $S^-$ the set of players $i$ with $\overline{K}_i > K_i^*$. We have

$$
\begin{aligned}
\sum_{i=1}^{m} z_i(K_i^*) - \sum_{i=1}^{m} z_i(\overline{K}_i) &= \sum_{i\in S^+} \big(z_i(K_i^*) - z_i(\overline{K}_i)\big) - \sum_{i\in S^-} \big(z_i(\overline{K}_i) - z_i(K_i^*)\big) && (2.6)\\
&= \sum_{i\in S^+} \sum_{j=\overline{K}_i+1}^{K_i^*} w_{ij} - \sum_{i\in S^-} \sum_{j=K_i^*+1}^{\overline{K}_i} w_{ij} && (2.7)\\
&\leq \sum_{i\in S^+} w_{i,\overline{K}_i+1}(K_i^* - \overline{K}_i) - \sum_{i\in S^-} w_{i,\overline{K}_i}(\overline{K}_i - K_i^*) && (2.8)\\
&= \sum_{i\in S^+} \Delta_i^+(K_i^* - \overline{K}_i) - \sum_{i\in S^-} \Delta_i^-(\overline{K}_i - K_i^*) && (2.9)\\
&\leq (\Delta^+ - \Delta^-) \sum_{i\in S^+} (K_i^* - \overline{K}_i) && (2.10)\\
&\leq 0.
\end{aligned}
$$

We partition the sum into two sets to get (2.6). The marginal changes $z_i(K_i + 1) - z_i(K_i)$ correspond to the weight $w_{i,K_i+1}$; and this substitution gives us (2.7). The order of the items imply $w_{i,K_i} \geq w_{i,K_i+1}$; and we use these bounds to get (2.8). According to the allocation $\overline{K}$, $w_{i,\overline{K}_i+1}$ is the weight of the first item to be picked by player $i$, which is $\Delta_i^+$; $w_{i,\overline{K}_i}$ is the weight of the first item to be dropped off, $\Delta_i^-$; hence (2.9). $\Delta_i^+$ is trivially upperbounded by $\Delta^+$ and $\Delta_i^-$ is lowerbounded by $\Delta^-$. Taking those two out of the summation, and the fact that $\sum_{i\in S^+}(K_i^* - \overline{K}_i) = \sum_{i\in S^-}(\overline{K}_i - K_i^*)$ yields (2.10); and since $\Delta^+ < \Delta^-$ at termination, this term is less than or equal to zero. This conclusion contradicts our assumption that $\sum_{i=1}^m z_i(K_i^*) > \sum_{i=1}^m z_i(\overline{K}_i)$; therefore no such $K^*$ exists, $\overline{K}$ is the optimal allocation.

Let $K'$ be the resource allocation used to initialize the CC-Lin algorithm. Consider the set $S^+ = \left\{i : \overline{K}_i > K_i'\right\}$ and a player $i \in S^+$. By the construction of our algorithm, $K_i$ increases by 1 unit only when player $i$ is the player with the highest gain and is $i^+$. Lemma 2.3 implies that player $i$ is $i^+$ at exactly $\overline{K}_i - K_i'$ iterations. The algorithm allows only 1 winner at each iteration; therefore it requires $\sum_{i\in S^+} \overline{K}_i - K_i'$ iterations for players in $i \in S^+$ to reach resource level $\overline{K}_i$. A similar argument can be made for players in

$S^- = \{i : \overline{K}_i < K'_i\}$; and it requires $\sum_{i \in S^-} K'_i - \overline{K}_i$ iterations for them to reach $\overline{K}_i$. Finally, the players in neither of these sets cannot be $i^+$ or $i^-$ at any iteration, by Lemma 2.3. Notice that the two processes, selecting $i^+$ or $i^-$, are performed concurrently; therefore the algorithm terminates in $\sum_{i \in S^-} K'_i - \overline{K}_i = \sum_{i \in S^+} \overline{K}_i - K'_i \leq K$ iterations. □

*Communication Complexity*

In the CC-Lin algorithm, the players need to share two pieces of information with each other: how much an additional unit of resource is worth to them ($\Delta_i^+$) and how much one less unit of resource will cost them($\Delta_i^-$). The function $BROADCAST()$ is where each player communicates this information with the others. In the first iteration, $BROADCAST()$ is called $m$ times (once for each player), where all processors broadcast to all processors. This communication scheme is called *Multinode broadcast*. In the remaining iterations $BROADCAST()$ is only called twice (only by players $i^+$ and $i^-$). In that case the type of communication is called *Single node accumulation*. Optimal communication times for multinode broadcast and single node accumulation in various network structures are given in Table 2.1.

Table 2.1: Solution times of optimal algorithms for the basic communication problems using a ring, a binary balanced tree, and a $d$-dimensional symmetric mesh with $m$ processors [22]

| | Ring | Tree | Mesh |
|---|---|---|---|
| Single node accumulation | $\Theta(m)$ | $\Theta(\log m)$ | $\Theta(m^{1/d})$ |
| Multinode broadcast | $\Theta(m)$ | $\Theta(m)$ | $\Theta(m)$ |

CC-Lin as described exploits the memory of processors in order to minimize communication by storing the $\Delta_i^+$ and $\Delta_i^-$ values of each player. Other communication designs are also possible. Given a communication topology in the underlying network, in two rounds of communication the same goal can be achieved; namely in the first round $\Delta_i^+$, $i^+$, $\Delta_i^-$, $i^-$ are computed, and in the second round this information is broadcasted to all players. The advantage of this communication scheme is the lack of memory requirement, and the

disadvantage is that every player has to share and receive data at all iterations.

### 2.3.2 The CC-Log Algorithm

The next algorithm that we propose is based on the distributed algorithm for finding the median in [23]. Consider the setting in (2.3) with only two players, $m = 2$, where without loss of generality each player has $K$ items with corresponding item values, i.e. $N_i = K$; and the task is to find the median of the item values. Given an array $U$ of size $n$ of non-increasing values, let $median1(U)$ be the $\lfloor \frac{n}{2} \rfloor^{th}$ element. Notice that for all optimal solutions $x^*$ of (2.3), if $x^*_{ij} = 1$, then $w^*_{ij} \geq median1(w)$ and if $x^*_{ij} = 0$, then $w^*_{ij} \leq median1(w)$. In other words, for an optimal allocation $\{K^*_1, K^*_2\}$, $median1(w) = \min\left\{w_{1,K^*_1}, w_{2,K^*_2}\right\}$.

Bearing in mind the similarities between our problem (2.3) and the problem of finding the median, we propose the following variant of the algorithm in [23]. Players 1 and 2 both start with zero initial resource allocation, i.e. $K_i = 0$; and the remaining resource, $\overline{K}$, is initially $K$. They share one piece of information with each other, $\Delta_i$: The value of the next best item they would select assuming they have half of the remaining resource. Namely $\Delta_i = z_i(K_i + \lfloor \frac{\overline{K}}{2} \rfloor + 1) - z_i(K_i + \lfloor \frac{\overline{K}}{2} \rfloor)$. This value can be interpreted as the median of the values of the unselected items of player $i$. Let $i^+ = argmax\{\Delta_i\}$. Player $i^+$ gets half of the remaining resource, and the remaining resource is reduced by half. This process is repeated until the remaining resource is zero. We refer to this decentralized algorithm as CC-Log, short for *cardinality constraint logarithmic convergence*, and describe it in Figure 2.2.

CC-Log is shown to terminate in $\log K$ iterations [23], as at each iteration $\overline{K}$ is reduced by half. The optimality for the two player setting comes from an inductive proof using Theorem 2.2.

**Theorem 2.2.** *Let* $\Delta_i = z_i(\lfloor \frac{\overline{K}}{2} \rfloor + 1) - z_i(\lfloor \frac{\overline{K}}{2} \rfloor)$ *for* $i = 1, 2$. *If* $\Delta_1 \geq \Delta_2$, *then there exists an optimal resouce allocation* $\{K^*_1, K^*_2\}$ *that satisfies* $K^*_1 \geq \lceil \frac{\overline{K}}{2} \rceil$.

*Proof.* Assume otherwise. Namely, for all optimal resource allocations assume $K_1 < \lceil \frac{\overline{K}}{2} \rceil$

**Input:** Player index $i$, total resource $K$
**Output:** Resource allocation $K_i$

**while** $\overline{K} > 0$ **do**
  $\Delta_i = z_i(K_i + \lfloor \frac{\overline{K}}{2} \rfloor + 1) - z_i(K_i + \lfloor \frac{\overline{K}}{2} \rfloor) + \epsilon(i)$
  BROADCAST($\Delta_i$)
  RECEIVE($\Delta_{i'}$)
  **if** $\Delta_i > \Delta_{i'}$ **then**
    $K_i = K_i + \lceil \frac{\overline{K}}{2} \rceil$
  **end if**
  $\overline{K} = \overline{K} - \lceil \frac{\overline{K}}{2} \rceil$
**end while**

Figure 2.2: The CC-Log algorithm *(for 2 players)*

holds. Let $\{K_1^*, K_2^*\}$ be an optimal resource allocation with maximum $K_1^*$. Define $z^* = z_1(K_1^*) + z_2(K_2^*)$ and $\bar{z} = z_1(K_1^* + 1) + z_2(K_2^* - 1)$. We have

$$\begin{aligned} \bar{z} &= z^* + z_1(K_1^* + 1) - z_1(K_1^*) + z_2(K_2^*) - z_2(K_2^* - 1) \\ &\geq z^* + \Delta_1 - \Delta_2 \\ &\geq z^*. \end{aligned}$$

Notice that since $K_1^* < \lceil \frac{\overline{K}}{2} \rceil$, the value of the $(K_1^* + 1)^{st}$ item of player 1, $z_1(K_1^* + 1) - z_1(K_1^*)$, is at least the value of the $(\lfloor \frac{\overline{K}}{2} \rfloor + 1)^{th}$ item, $\Delta_1$. Similarly, the value of the $K_2^{*th}$ item of player 2, $z_2(K_2^*) - z_2(K_2^* - 1)$, is at most $\Delta_2$. By our assumption we have $\Delta_1 \geq \Delta_2$; therefore $\bar{z} \geq z^*$ is implied. Either $\bar{z} > z^*$ contradicting the optimality of $\{K_1^*, K_2^*\}$ , or $\bar{z} = z^*$ meaning $\{K_1^* + 1, K_2^* - 1\}$ is an optimal solution with higher resource allocated to player 1, contradicting our assumption. □

**Theorem 2.3.** *The CC-Log algorithm described in Figure 2.2 outputs an optimal resource allocation for problem (2.3) for* $m = 2$.

*Proof.* Each iteration of the CC-Log algorithm can be interpreted as discarding half of the feasible region of resource allocations, until we get a single point as the remaining

feasible region. Theorem 2.2 indicates that there exists at least one optimal solution in the remaining feasible region after each iteration. Therefore, inductively we can conclude that the last feasible solution remaining is an optimal solution. □

We generalize the CC-Log algorithm described in Figure 2.2 to a multiplayer setting by partitioning the players into two groups, having the union of the players in a group act as a single entity, and thereby imitate the two player game. The new algorithm, also called CC-Log, is described in Figure 2.3.

**Input:** Set of players $I \subseteq \{1 \dots m\}$, total resource $K$
**Output:** Resource allocation $K_I$
1: **while** $\overline{K} > 0$ **do**
2: $\quad \Delta_I = z_I(K_I + \lfloor \frac{\overline{K}}{2} \rfloor + 1) - z_I(K_I + \lfloor \frac{\overline{K}}{2} \rfloor) + \epsilon(I)$
3: $\quad$ BROADCAST($\Delta_I$)
4: $\quad$ RECEIVE($\Delta_{\bar{I}}$)
5: $\quad$ **if** $\Delta_I > \Delta_{\bar{I}}$ **then**
6: $\quad\quad K_I = K_I + \lceil \frac{\overline{K}}{2} \rceil$
7: $\quad$ **end if**
8: $\quad \overline{K} = \overline{K} - \lceil \frac{\overline{K}}{2} \rceil$
9: **end while**

Figure 2.3: The CC-Log algorithm *(for multiple players)*

It can easily be seen that the CC-Log algorithm in Figure 2.3 is structually equivalent to the CC-Log algorithm in Figure 2.2. The main challange of the CC-Log algorithm in Figure 2.3 is Line 2, where

$$z_I(K_I) = \max \left\{ \sum_{i \in I} w_i^T x_i : \sum_{i \in I} \mathbf{1}^T x_i \leq K_I,\ x_i \in \{0,1\}^{N_i}\ \ i \in I \right\}. \qquad (P_I)$$

In other words, $z_I(K_I)$ is the resulting objective value when the set of players $I$ allocate $K_I$ units of resource optimally among each other. Notice that our main problem (2.3) is a special case of $(P_I)$ where $I = \{1, \dots, m\}$ and $K_I = K$. Therefore solving $(P_I)$ is at least as hard as solving the original problem (2.3). We propose to solve $(P_I)$ using the same approach that we proposed for (2.3): the CC-Log algorithm. This is a nested algorithm. At every iteration of the CC-Log($I$), in order to compute $z_I(\cdot)$ the group of players $I$ run the

algorithm CC-Log($I'$) against CC-Log($I \backslash I'$) for some $\emptyset \subset I' \subset I$. Unless $|I| = 2$, at least one of the sets $I'$ and $I \backslash I'$ has size strictly greater than one; and hence CC-Log called for that set is Figure 2.3 again. The recursion stops when the set is a singleton and the CC-Log function called is Figure 2.2.

*Convergence and Optimality*

In order to analyze the convergence of the algorithm, we will construct the following game tree. Every node $v$ has a non-empty group of players associated with it, $I_v \subseteq \{1, \ldots, m\}$. For the root node $r$, we have $I_r = \{1, \ldots, m\}$. A node $v$ has two child nodes $v_1$ and $v_2$, unless $|I_v| = 1$, in which case $v$ is a leaf node. The set of players associated with $v$ is partitioned to $v_1$ and $v_2$, i.e. $I_v = I_{v_1} \cup I_{v_2}$ and $I_{v_1} \cap I_{v_2} = \emptyset$. We use this game tree to model how the set of players is partitioned into groups to compute $z_I(\cdot)$ and run the decentralized algorithm CC-Log. Our assumption is that the two siblings $v_1$ and $v_2$ run CC-Log against each other to compute $z_{I_v}(\cdot)$ for their parent $v$. Note that all nodes except the root node have one sibling and one parent.

Given such a game tree, the two children of the root node $r$ start by playing the decentralized game against each other; where we have CC-Log($I_{r_1}$) against CC-Log($I_{r_2}$). At every iteration of the algorithm CC-Log($I_v$), in order to compute $z_{I_v}(\cdot)$, node $v$ starts a new game between the two children CC-Log($I_{v_1}$) against CC-Log($I_{v_2}$), unless $|I_v| = 1$. Let $\gamma(v, K_{I_v})$ be the number of iterations CC-Log requires to find $z_{I_v}(K_{I_v})$. In other words $\gamma(v, K_{I_v})$ is the number of iterations required for the game CC-Log($I_{v_1}$) against CC-Log($I_{v_2}$) to converge with initial remaining resource $\overline{K} = K_{I_v}$, where $v_1$ and $v_2$ are the children of $v$. In the case where $|I_v| = 1$ and $v_1$ and $v_2$ don't exist, we assume $\gamma(v, K_{I_v}) = 1$. Recall that as explained in [23], the while loop in CC-Log takes $\log K_{I_v}$ iterations; and at every iteration we call CC-Log for both of the child nodes with $\overline{K}$ values at most $K_{I_v}$. Thus we have the

recursion:

$$\gamma(v, K_{I_v}) \leq \log K_{I_v} \cdot \max\left\{\gamma(v_1, K_{I_v}), \gamma(v_2, K_{I_v})\right\}. \tag{2.11}$$

**Theorem 2.4.** *The CC-Log algorithm applied to (2.3) terminates in at most* $(\log K)^{\alpha}$ *iterations, where* $\alpha$ *varies between* $\log m$ *and* $m$ *depending on the communication network structure.*

*Proof.* The value we want to bound to prove Theorem 2.4 is $\gamma(r, K)$. Notice that the bound (2.11) multiplies by a factor of $\log K$ with each level of children until we reach a node with both children as leaf nodes, in which case $\max\left\{\gamma(v_1, K), \gamma(v_2, K)\right\} = 1$. Therefore, the bound (2.11) for $\gamma(r, K)$ becomes $(\log K)^{\alpha}$, where $\alpha$ is the depth of the game tree. The structure of the tree highly depends on the communication network among the players; i.e. which players can communicate directly with each other and can act as a single group in the CC-Log algorithm in Figure 2.3. For a perfectly balanced game tree as the best case scenario, i.e. a tree with leaf nodes having almost the same depth, we have $\alpha = \log m$. For a skewed tree as the worst case scenario, i.e. a tree with almost all leaf nodes having different depths, we have $\alpha = m - 1$.

□

For the rest of the chapter, we will assume a complete communication network, and a balanced game tree, and hence a $(\log K)^{\log m}$ upper bound on the convergence of the CC-Log algorithm. Although the impact of $K$ on the convergence has been reduced to $\log K$, now the convergence bound also involves $m$. Thus neither the CC-Lin or the CC-Log algorithm dominates the other.

**Theorem 2.5.** *The CC-Log algorithm applied to (2.3) outputs an optimal resource allocation.*

*Proof.* Recall that given a game tree, for a node $v$ our algorithm solves for $z_{I_v}(\cdot)$ by having the two child nodes $v_1$ and $v_2$ play CC-Log against each other. The main goal is to show the

optimality of $z_{I_r}(K)$, which we will prove by induction on $|I_v|$. For a node $v$ with $|I_v| = 1$, $z_{I_v}(\cdot)$ is computed optimally as it is the player's individual problem (2.4) for some player $i$.

Now assume $|I_v| = \alpha$ for some $\alpha > 2$ and that $z_I(\cdot)$ is computed optimally for all sets $I$ with $|I| < \alpha$. In order to compute $z_{I_v}(\cdot)$, the child nodes play CC-Log($I_{v_1}$) against CC-Log($I_{v_2}$). Let $i_1$ and $i_2$ be two dummy players who have complete information on the problems of players $I_{v_1}$ and $I_{v_2}$ respectively, and define $z_{i_j}(\cdot)$ to be the optimal solution of the problem $(P_I)$ for $I = I_{v_j}$ for $j = 1, 2$. By our induction, since $|I_{v_j}| < |I_v| = \alpha$ we assume that $z_{I_{v_j}}(\cdot)$ computed by multiplayer CC-Log is optimal, i.e. $z_{I_{v_j}}(\cdot)$ used in Figure 2.3 by $v_j$ is equal to $z_{i_j}(\cdot)$ for $j = 1, 2$. The information shared in the two-player Figure 2.2 between $i_1$ and $i_2$ is identical to the information shared in the multiplayer Figure 2.3 between $I_{v_1}$ and $I_{v_2}$. Therefore the resource allocation decisions at each iteration and the output of the two-player CC-Log algorithm between $i_1$ and $i_2$ equals resource allocation decisions at each iteration and the output of the multiplayer CC-Log algorithm between $I_{v_1}$ and $I_{v_2}$. The optimality guarantee of the two-player game in Theorem 2.3 holds for the multiplayer game; and hence $z_{I_v}$ is computed optimally. □

## 2.4 Generalizing the Cardinality Constraint Setting by Accomodating Local Constraints

The decentralized algorithms CC-Lin and CC-Log are introduced and analyzed on a very simple item selection problem; however, their applications are not restricted to it. In this section we provide a sufficient condition of optimality for more general problem types and give examples of problems that satisfy this sufficient condition.

Given the generalized problem (2.1) and the players' individual problems (2.2), con-

struct the problem:

$$\widetilde{z} = \max \left\{ \sum_{i=1}^{m} \widetilde{w}_i^T y_i \mid \sum_{i=1}^{m} \mathbf{1}^T y_i \leq K,\ y_i \in \{0,1\}^{N_i}\ \ i = 1, \ldots, m \right\} \tag{2.12}$$

$$\widetilde{z}_i(K_i) = \max \left\{ \widetilde{w}_i^T y_i \mid \mathbf{1}^T y_i \leq K_i,\ y_i \in \{0,1\}^{N_i} \right\} \tag{2.13}$$

where $N_i = K$ for all players $i = 1 \ldots m$, and the weights $\widetilde{w}_{ij} = z_i(j) - z_i(j-1)$ for $j = 1 \ldots K$, where $z_i(\cdot)$ is the optimal objective function value of problem (2.2). In other words, we treat the marginal changes in the objective function value with each additional unit of resource as an item, where the value of the item equals the value of the marginal increase. Notice that (2.12) is structually equivalent to (2.3), and has the convergence properties discussed in Sect. 2.3.

**Definition 2.1** (Concavity Property)**.** The problem (2.2) is defined to have the *concavity property* when $z_i(K_i + 2) - z_i(K_i + 1) \leq z_i(K_i + 1) - z_i(K_i)$ , $z_i(0) = 0$ and $z_i(K_i) = z_i(K_i + \epsilon)$ for all integer $K_i$ and $\epsilon \in [0, 1)$.

In other words, if problem (2.2) has the concavity property, then the marginal utility of an additional unit of resource for player $i$ is non-increasing. The concavity property of a player's individual problem (2.2) implies that the player's items in problem (2.13) are ordered; namely $\widetilde{w}_{i,j} \geq \widetilde{w}_{i,j+1}$. This observation leads us to the conclusion that $z_i(K_i) = \widetilde{z}_i(K_i)$ for all integer $K_i = 0, \ldots, K$ because

$$\begin{aligned} \widetilde{z}_i(K_i) &= \sum_{j=1}^{K_i} \widetilde{w}_{ij} && (2.14) \\ &= \sum_{j=1}^{K_i} (z_i(j) - z_i(j-1)) && (2.15) \\ &= z_i(K_i). && (2.16) \end{aligned}$$

Here, (2.14) is the optimal solution of the cardinality knapsack problem (2.13), as the $K_i$ most valuable items are items $j = 1, \ldots, K_i$ by the concavity property. (2.15) is by the definition of $\widetilde{w}_{ij}$, and we get (2.16) by simplification. Furthermore, due to our initial assumption of all data being integer, the objective function $z_i(K_i)$ of (2.2) is a discrete function of $K_i$; namely for any $\epsilon \in [0, 1)$ and $K_i \in \mathbb{Z}_+$, $z_i(K_i) = z_i(K_i + \epsilon)$. With this observation, it suffices to show the equivalence of $z_i$ and $\widetilde{z}_i$ for integer $K_i$ values. When the objective functions behave identically, the information each player shares, $\Delta_i^+$ and $\Delta_i^-$ in CC-Lin and $\Delta_i$ in CC-Log, is the same whether they are solving problem (2.2) or (2.13); which implies that the reassignment of the resources is also the same. Hence, when problems (2.13) have the concavity property for all players $i = 1, \ldots, m$, the steps of CC-Lin and CC-Log when solving (2.1) are the same as when solving (2.12). All the convergence properties that hold for (2.13), i.e. a single cardinality knapsack problem as discussed in Sect. 2.3, also holds for (2.1) when (2.2) has the concavity property for all players $i = 1, \ldots, m$. Namely, CC-Lin outputs the optimal solution in at most $K$ iterations, and CC-Log outputs the optimal solution in at most $(\log K)^{\log m}$ iterations.

### 2.4.1 Example Problems with the Concavity Property

In order to prove concavity, we restrict ourselves to problems (2.1) that can be reformulated into the following form:

$$z = \max \left\{ \sum_{i=1}^{m} w_i^T x_i : \sum_{i=1}^{m} \mathbf{1}^T x_i \leq K \, , \; x_i \in X_i \, , \; x_i \in \{0,1\}^{N_i} \quad i = 1, \ldots, m \right\}. \tag{2.17}$$

Along with the cardinality constraint, each player has another set of constraints it needs to satisfy, which are described by $x_i \in X_i$. When we distribute (2.17) into each player's

individual subproblems, we get

$$z_i(K_i) = \max \left\{ w_i^T x_i : \mathbf{1}^T x_i \leq K_i \ , \ x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \tag{2.18}$$

**Theorem 2.6** (Aghezzaf and Wolsey [24]). *Given a set of feasible solutions $X_i$, and the convex hull of $X_i$, $conv(X_i)$, is integral, i.e. all of its extreme points are integer, if the following holds for all nonnegative integer $k$ values, then (2.18) has the concavity property:*

$$conv(X_i) \cap \left\{ x : \sum_{j=1}^{N_i} x_{ij} = k \right\} = conv \left( X_i \cap \left\{ \sum_{j=1}^{N_i} x_{ij} = k \right\} \right).$$

Notice that for pure integer problems, this is equivalent to $conv \left( X_i \cap \left\{ x : \sum_{j=1}^{N_i} x_{ij} = k \right\} \right)$ being an integer polytope. For the problems listed below, there are results in the literature indicating that they have the concavity property. Thus for any of these problems with a cardinality constraint formulated as (2.18), all of the results of Section 2.3 are valid.

- **Matroid:** Given a ground set of elements $N = \{1, \ldots, n\}$ and a rank function $r(\cdot)$ on the ground set that satisfies $r(\cdot) \geq 0$, $r(S) \leq |S|$ for all $S \subseteq N$, $r(S) + r(T) \geq r(S \cap T) + r(S \cup T)$ for $S, T \subseteq N$, and $r(S) \leq r(T) \leq r(N)$ for all $S \subset T \subset N$, a matroid polytope is defined as

  $$X = \left\{ x : \sum_{j \in S} x_j \leq r(S) \ \ \forall S \subseteq N \ , \ x \in \{0,1\}^n \right\}.$$

  Since the cardinality constraint itself represents a matroid, the intersection of the matroid constraint with the cardinality constraint becomes the intersection of two matroid polytopes. The integrality result of the intersection of two matroid polytopes [25] implies the concavity of the matroid problem.

- **Intersection of two matroids:** Given a ground set of elements $N = \{1, \ldots, n\}$ and two rank functions $r_1(\cdot)$ and $r_2(\cdot)$ on the ground set, the intersection of two matroid

polytopes is defined as

$$X = \left\{ x : \sum_{j \in S} x_j \leq r_1(S) \ \ \forall S \subseteq N \ , \ \sum_{j \in S} x_j \leq r_2(S) \ \ \forall S \subseteq N \ , \ x \in \{0,1\}^n \right\}.$$

It is shown in [25] that $conv(X) \cap \left\{ \sum_{e \in E} x_e = k \right\}$ is an integral polytope.

- **Matching:** Given a graph $G = (V, E)$ and a $\delta(\cdot)$ function defined on the set $V$ as $\delta(v) = \{e = (u, v) : e \in E\}$, the set of feasible matchings is defined as

$$X = \left\{ x : \sum_{e \in \delta(v)} x_e \leq 1 \ \ \forall v \in V \ , \ x \in \{0,1\}^{|E|} \right\}.$$

It is shown in [26] that $conv(X) \cap \left\{ x : \sum_{e \in E} x_e = k \right\} = conv(X \cap \left\{ x : \sum_{e \in E} x_e = k \right\})$.

- **Integer $b$-matching:** Given a graph $G = (V, E)$ and an integer vector $b \in \mathbf{Z}_+^{|V|}$, the set of feasible integer $b$-matchings are defined as:

$$X = \left\{ x : \sum_{e \in \delta(v)} x_e \leq b(v) \ \ \forall v \in V \ , \ x \in \mathbf{Z}_+^{|E|} \right\}.$$

To show that integer $b$-matching has the concavity property construct the graph $G' = (V', E')$ in the following manner: For all vertices $v \in V$, there are $b(v)$ copies in $V'$. For all edges $(u, v) \in E$, there exists an edge in $E'$ between all copies of $u$ and $v$. Let $X'$ be the set of feasible binary matchings in $G'$:

$$X' = \left\{ x' : \sum_{e \in \delta(v)} x'_e \leq 1 \ \ \forall v \in V' \ , \ x' \in \{0,1\}^{|E'|} \right\}.$$

Clearly, any feasible integer $b$-matching $x \in X$ can be broken down into a binary vector $x' \in X'$, and similarly any feasible binary matching $x' \in X'$ can be compres-

sed to an integer vector $x \in X$. This transformation preserves the sum of elements in the vector, namely $\mathbf{1}^T x = \mathbf{1}^T x'$. Furthermore, given a vector of objective coefficients $c \in \mathbf{Z}_+^{|E|}$, we will let $c' \in \mathbf{Z}_+^{|E'|}$ be such that $c(e) = c'(e')$ for any $e = (u, v)$ and $e' = (u', v')$ where $u'$ and $v'$ are copies of $u$ and $v$ respectively. Notice that for any two vectors $x \in X$ and $x' \in X'$ that are equivalent, we have $c^T x = c'^T x'$. For any given vector $c$, solving the integer $b$-matching problem $\max\left\{c^T x | x \in X\right\}$ is equivalent to solving the binary matching problem $\max\left\{c'^T x' | x' \in X'\right\}$. This reduction is explained in more detail in [25]. Moreover, because the transformation between the equivalent vectors $x \in X$ and $x' \in X'$ preserves cardinality, we can conclude that solving the integer $b$-matching problem $\max\left\{c^T x | x \in X\ ,\ \mathbf{1}^T x = k\right\}$ is equivalent to solving the binary matching problem $\max\left\{c'^T x' | x' \in X'\ ,\ \mathbf{1}^T x' = k\right\}$. We know that the binary matching problem has the concavity property. Since the objective function of these two problems match, we can conclude that the integer $b$-matching problem also has the concavity property.

- **Binary $b$-matching:** Given a graph $G = (V, E)$ and an integer vector $b \in \mathbf{Z}_+^{|V|}$, the set of feasible binary $b$-matchings are defined as:

$$X = \left\{ x : \sum_{e \in \delta(v)} x_e \leq b(v)\ \ \forall v \in V\ ,\ x \in \{0,1\}^{|E|} \right\}.$$

To show that binary $b$-matching has the concavity property construct the graph $G' = (V', E')$ in the following manner, as described in [26]: For all edges $e = (u, v) \in E$, construct two vertices $p_{eu}, p_{ev} \in \bar{V}$. The vertex set $V'$ is defined as $V \cup \bar{V}$. The edge set is defined as for all edges $e = (u, v) \in E$, the edges $(u, p_{eu}), (v, p_{ev}), (p_{eu}, p_{ev})$ are in $E'$. The weights for all these three edges $e'$, $w'_{e'}$, equals the weight of the original edge, $w_e$, and $b'(v) = b(v)$ for all $v \in V$, and $b'(v) = 1$ for all $v \in \bar{V}$. Consider the integer $b'$-matching problem defined on $G' = (V', E')$ using $w'$ as the objective coefficients. Notice that because of the construction of the graph $G'$, the value of each

edge is upper bounded by 1 even in the integer $b'$-matching problem. Let $x$ be a binary $b$-matching on $G$. We construct $x'$ as a feasible solution to the integer $b'$-matching problem on $G'$ such that for all $e = (u, v)$, we let $x_e = x'_{u,p_{eu}} = x'_{v,p_{ev}} = 1 - x'_{p_{eu},p_{ev}}$. The value of the solution $x$ equals $\sum_{e\in E} w_e x_e$, and the value of the solution $x'$ equals $\sum_{e\in E}(w_e x_e + w_e) = W + \sum_{e\in E} w_e x_e$, where $W$ is the constant $\sum_{e\in E} w_e$. Furthermore, the cardinality of $x'$ equals $\sum_{e\in E}(1 + x_e) = |E| + \sum_{e\in E} x_e$. Now let's do the reverse construction, let $x'$ be a maximum weight integer $b'$-matching on $G'$. Notice that for any $e = (u, v) \in E$, the sum $x'_{u,p_{ue}} + x'_{v,p_{ve}} + x'_{p_{ue},p_{ve}}$ equals either 1 or 2, simply because the sum equals 0 is suboptimal, and the sum equals 3 is not feasible. For $e = (u, v) \in E$ such that the sum equals 1, we set $x_e = 0$, and if the sum equals 2, we set $x_e = 1$. Notice that the sum can only equal 2 if $x'_{u,p_{eu}} = x'_{v,p_{ev}} = 1$, therefore setting $x_e = 1$ does not violate the capacity constraints on nodes $u$ or $v$. By similar arguments, we can conclude that $\sum_{e\in E'} w'_e x'_e = W + \sum_{e\in E} w_e x_e$ and $\sum_{e\in E'} x'_e = |E| + \sum_{e\in E} x_e$. Because of this constant difference between the cardinalities and the objective function values, we reach the following conclusion: Given a graph $G = (V, E)$, an integer vector $b \in \mathbf{Z}_+^{|V|}$, and objective function coefficients $w \in \mathbf{R}_+^{|E|}$, the value of a maximum weight binary $b$-matching on $G$ of cardinality at most $k$ is $W$ less than the value of a maximum weight integer $b'$-matching on $G'$ of cardinality at most $k + |E|$. Because of this equivalence, the concavity of the integer $b$-matching problem implies the concavity of the binary $b$-matching problem.

- **Transportation problem with market choice:** Given a bipartite graph $G = (V_1 \cup V_2, E)$, where $V_1$ denotes the supply nodes, $V_2$ denotes the demand nodes, and $E$ is the available transportation links among them, the set $X$ defined as follows is the set of feasible solutions:

$$X = \left\{ (x,y) : \sum_{i:(i,j)\in E} x_{ij} = d_j(1-y_j) \ \ \forall j \in V_2 \ , \ \sum_{j:(i,j)\in E} x_{ij} \leq s_i \ \ \forall i \in V_1 \ , \right.$$
$$\left. y \in \{0,1\}^{|V_2|} \ , \ x \in \mathbf{R}_+^{|E|} \right\}.$$

Walter et.al.'s [27] result states that if $d_j \leq 2$ for all $j \in V_2$, then $conv(X) \cap \left\{(x,y) : \sum_{j\in V_2} y_j = k\right\}$ equals $conv\left(X \cap \left\{(x,y) : \sum_{j\in V_2} y_j = k\right\}\right)$.

The examples with concavity property that we have in the literature have some common properties. Namely, they all correspond to independence systems and the convex hull of the feasible regions described as $Ax \leq b$ have $A \geq 0$. However, both of these properties are not sufficient for concavity. A simple counter-example to an independence system that does not have the concavity property is the stable set problem. As for the nonnegativity of the polytope, consider the polytope $X = \{x_1 + x_2 \leq 1 \ , \ x_1 + x_3 \leq 1 \ , \ x_1 + x_4 \leq 1 \ , \ x \in [0,1]^4\}$, which is integral. $\left\{X \cap \mathbf{1}^T x = 2\right\}$ has a fractional extreme point, $(1/2, 1/2, 1/2, 1/2)$. Another integer polytope that does not satisfy the concavity property is the flow polytope, given by

$$X = \left\{ x : \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = 0 \ \ \forall i \in V \ , \ x \in \{0,1\}^{|E|} \right\}.$$

Notice that the flow polytope satisties neither of these properties, which leads to the question of the necessity of the two properties. This is still an open problem.

It should be noted that for the sake of the optimality of our algorithm, integrality of the polytope $conv\left(X_i \cap \left\{\sum_{j=1}^{N_i} x_{ij} = k\right\}\right)$ as a whole is not necessary. Aghezzaf and Wolsey [24] show that the uncapacitated lot sizing polytope intersected with a cardinality constraint on the binary variables defined as

$$X = \Bigg\{ (x,y) : \sum_{i=1}^{t} x_i \geq \sum_{i=1}^{t} d_i \;\; t = 1, \dots n-1 \;, \;\; \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} d_i \;, \\ 0 \leq x \leq My \;,\; y \in \{0,1\}^n \} \quad (2.19)$$

is integral along the direction $c$ on the continuous variables $x$, where $c_t \geq c_{t+1}$ for $t = 1, \dots n-1$. Hence the uncapacitated lot sizing problem has concavity property for specific family of objective coefficients.

## 2.5 Distributed Problems with a Cardinality Objective

Particularly in areas related to service industry where the goal is to maximize the number of customers served or the service centers to be placed, we come across problems described as

$$z = \max \left\{ \sum_{i=1}^{m} \mathbf{1}^T x_i : \sum_{i=1}^{m} a_i^T x_i \leq K,\; x_i \in \{0,1\}^{N_i} \;\; i = 1, \dots, m \right\}. \quad (2.20)$$

This is a knapsack problem with arbitrary item sizes, and the objective is maximizing the number of items selected. Again, the items are distributed among $m$ players, and the $i^{\text{th}}$ player is assigned $N_i$ items to choose from. The decision variable representing whether or not player $i$ chooses its $j^{\text{th}}$ item is $x_{ij}$, and the size of that item is $a_{ij}$. The players are expected to coordinate with each other in order to select the maximum number of items and not exceed the total capacity $K$.

Once the resource level $K$ is distributed among players, and player $i$ gets $K_i$ units of resource allocated, each player gets their individual problem (2.21):

$$z_i(K_i) = \max \left\{ \mathbf{1}^T x_i : a_i^T x_i \leq K_i \;,\; x_i \in \{0,1\}^{N_i} \right\}. \quad (2.21)$$

Without loss of generality, we assume the indices of $x_i$ are in increasing order of their item sizes; i.e. $a_{ij} \leq a_{i,j+1}$ for $j = 1 \ldots N_i - 1$. The optimal solution $x_i^*$ of (2.21) has $x_{ij}^* = 1$ for $j = 1, \ldots, j_i^*$ and $x_{ij}^* = 0$ for $j = j_i^* + 1, \ldots, N_i$, where $j_i^*$ satisfies $\sum_{j=1}^{j_i^*} a_{ij} \leq K_i$ and $\sum_{j=1}^{j_i^*+1} a_{ij} > K_i$. In our algorithm we will refer to the critical item $j_i^*$ as the player's threshold item index; and assume that $x_{ij}^* = 1$ for $j = 1, \ldots, j_i^*$ and $x_{ij}^* = 0$ for $j = j_i^* + 1, \ldots, N_i$ for an optimal solution $x^*$.

The algorithm works as follows: Each player is given an initial resource level $K_i$ that satisfies $\sum_{i=1}^{m} K_i = K$. The first thing they do is to calculate how much of this allocated resource they are using, by determining the threshold item index $j_i^*$. They update their resource level $K_i$ to the amount used, $\sum_{j=1}^{j_i^*} a_{ij}$; and return the unused amount $K_i - \sum_{j=1}^{j_i^*} a_{ij}$ to a slack pool among all players. At every iteration, players share two pieces of information with each other: the size of the next smallest item they will select, $\Delta_i^+$, which corresponds to $a_{i,j_i^*+1}$, and the size of the first item they will drop, $\Delta_i^-$, which corresponds to $a_{i,j_i^*}$. Once that information becomes public, every player computes the player with the smallest item to pick ($i^+$) and the player with the largest item to drop ($i^-$). First, they check whether the size of the smallest item to be picked is at most the slack. If so, player $i^+$ gets additional resource from the slack pool in the amount equal to the size of its candidate item. Otherwise, if the slack is not sufficient, they check whether the size of the smallest item to be picked is strictly smaller than the size of the largest item to be dropped. If so, player $i^-$ drops its candidate item, player $i^+$ picks its candidate item; and the slack pool is increased by the difference of the two item sizes. This process continues until the slack pool is not sufficient to pick another item, and the size of the smallest item to be picked is not smaller than the size of the largest item to be dropped; and at that point the algorithm terminates. Notice that after each iteration, all players except for either $i^+$, or $i^+$ and $i^-$ remain at the same resource level, and only the players with the new resource levels need to update and share their $\Delta_i^+$ and $\Delta_i^-$ values. We refer to this decentralized algorithm as CO-Lin, short for *cardinality objective linear convergence*, and describe it in Figure 2.4.

**Input:** Player index $i \in \{1 \dots m\}$, initial resource allocation $K_i$
**Output:** Resource allocation $K_i$

1: isChanged == **true**
2: Find $j_i^* :\ \sum_{j=1}^{j_i^*} a_{ij} \leq K_i$ and $\sum_{j=1}^{j_i^*+1} a_{ij} > K_i$
3: $S_i = K_i - \sum_{j=1}^{j_i^*} a_{ij}$ ; $K_i = \sum_{j=1}^{j_i^*} a_{ij}$
4: BROADCAST$(S_i)$
5: RECEIVE$\Big(\{S_p\}_{p\subseteq\{1,\dots,m\}}\Big)$
6: $S = \sum_{p=1}^{m} S_p$
7: **while true do**
8: **if** isChanged== **true then**
9: isChanged= **false**
10: $\Delta_i^+ = a_{i,j_i^*+1} + \epsilon(i)$ ; $\Delta_i^- = a_{i,j_i^*} + \epsilon(i)$
11: BROADCAST$(\Delta_i^+, \Delta_i^-)$
12: **end if**
13: RECEIVE$\Big(\{\Delta_p^+, \Delta_p^-\}_{p\subseteq\{1,\dots,m\}}\Big)$
14: $\Delta^+ = \min_{p\in\{1\dots m\}}\{\Delta_p^+\}$ ; $i^+ = arg\min_{p\in\{1\dots m\}}\{\Delta_p^+\}$
15: $\Delta^- = \max_{p\in\{1\dots m\}}\{\Delta_p^-\}$ ; $i^- = arg\max_{p\in\{1\dots m\}}\{\Delta_p^-\}$
16: **if** $S \geq \Delta^+$ **then**
17: $S = S - \Delta^+$
18: **if** $i^+ == i$ **then**
19: $K_i = K_i + \Delta^+$ , $j^* = j_i^* + 1$ , isChanged= **true**
20: **end if**
21: **else if** $\Delta^+ < \Delta^-$ **then**
22: $S = S + \Delta^- - \Delta^+$
23: **if** $i^+ == i$ **then**
24: $K_i = K_i + \Delta^+$ , $j_i^* = j_i^* + 1$ , isChanged= **true**
25: **else if** $i^- == i$ **then**
26: $K_i = K_i - \Delta^-$ , $j_i^* = j_i^* - 1$ , isChanged = **true**
27: **end if**
28: **else**
29: **return** $K_i$
30: **end if**
31: **end while**

Figure 2.4: The CO-Lin algorithm

### 2.5.1 Convergence and Optimality

In Line 10, as a tie-breaking rule the players add a function of their indices, $\epsilon(i)$, to the $\Delta_i^+$ and $\Delta_i^-$ values. An example for this function might be $\epsilon(i) = \varepsilon i$ for some $\varepsilon \approx 0$. $\epsilon(i)$ is significantly smaller in magnitude than $\Delta_i^+$ and $\Delta_i^-$; therefore it will not be included in the

convergence and optimality analyses for the sake of simplicity.

Our main result on the performance of the CO-Lin algorithm is stated in Theorem 2.7.

**Theorem 2.7.** *When the CO-Lin algorithm is applied to problem (2.20), it outputs an optimal resource allocation in at most* $N = \sum_{i=1}^{m} N_i$ *iterations.*

The following results are used to build up to the proof of Theorem 2.7.

**Lemma 2.4.** When the CO-Lin algorithm is applied to problem (2.20), resource exchange occurs between two distinct players.

*Proof.* Assume $i = i^+ = i^-$, with threshold item index $j_i^*$. $\Delta_i^+ = a_{i,j_i^*+1} = \Delta^+$ and $\Delta_i^- = a_{i,j_i^*} = \Delta^-$ hold by our assumption. As $a_{i,j_i^*+1} \geq a_{i,j_i^*}$, we have $\Delta^+ \geq \Delta^-$, which implies that the algorithm either uses the slack to increase a player's resource (satisfying the condition in Line 16), or it terminates (satisfying the condition in Line 28). In either case, a resource exchange between $i^+$ and $i^-$ does not occur when $i^+ = i^-$. □

For the following proofs, let the superscript $(t)$ refer to the value of a parameter at iteration $t$ of the algorithm.

**Lemma 2.5.** When the CO-Lin algorithm is applied to problem (2.20), the sequence of $\Delta^+$ values throughout the iterations are nondecreasing; and the sequence of $\Delta^-$ values are nonincreasing.

*Proof.* We will prove the first part of Lemma 2.5 by showing that $\Delta^{+(t+1)} \geq \Delta^{+(t)}$ for all $t$ values. The second part, $\Delta^{-(t+1)} \leq \Delta^{-(t)}$ follows from a similar argument. We have that

$$\Delta^{+(t)} = \min_{i=1,\ldots,m} \left\{ \Delta_i^{+(t)} \right\} = \Delta_{i^{+(t)}}^{+(t)}$$

holds by the definition of $\Delta^+$ and $i^+$, and this value equals $a_{i^{+(t)},j_{i^{+(t)}}^{*(t)}+1}$, the size of the next smallest item player $i^{+(t)}$ will pick, with respect to their current threshold level $j_{i^{+(t)}}^{*(t)}$. Since we assume the algorithm doesn't terminate at iteration $t$, it has two courses of action: either

$i^{+(t)}$ adds another item using the slack (Line 16); or $i^{+(t)}$ and $i^{-(t)}$ exchange resources (Line 21). In either case, we can write a similar definition of $\Delta^{+(t+1)}$ and expand it as (Sc1) and (Sc2) respectively for these two scenarios:

$$\Delta^{+(t+1)} = \min_{i=1,\dots,m} \left\{ \Delta_i^{+(t+1)} \right\} = \min \left\{ \min_{i \neq i^{+(t)}} \left\{ \Delta_i^{+(t)} \right\}, \Delta_{i^{+(t)}}^{+(t+1)} \right\} \tag{Sc1}$$

$$\Delta^{+(t+1)} = \min_{i=1,\dots,m} \left\{ \Delta_i^{+(t+1)} \right\} = \min \left\{ \min_{i \neq i^{+(t)}, i^{-(t)}} \left\{ \Delta_i^{+(t)} \right\}, \Delta_{i^{-(t)}}^{-(t)}, \Delta_{i^{+(t)}}^{+(t+1)} \right\} \tag{Sc2}$$

First notice that, after iteration $t$, only the players $i^{+(t)}$ and $i^{-(t)}$ may update their $\Delta_i$ values; therefore, except for those two players we have $\Delta_i^{+(t+1)} = \Delta_i^{+(t)}$. By the definition of $\Delta^+$ and $i^+$, $\min_{i \neq i^{+(t)}} \left\{ \Delta_i^{+(t)} \right\} \geq \Delta^{+(t)}$ for (Sc1) and $\min_{i \neq i^{+(t)}, i^{-(t)}} \left\{ \Delta_i^{+(t)} \right\} \geq \Delta^{+(t)}$ for (Sc2). Under the second scenario, the optimality conditions of player $i^{-(t)}$ imply that the size of the item dropped at iteration $t$ must be equal to the size of the next best item to be selected at iteration $t+1$; hence $\Delta_{i^{-(t)}}^{+(t+1)} = \Delta_{i^{-(t)}}^{-(t)}$. Assuming the second scenario, $\Delta_{i^{-(t)}}^{-(t)} = \Delta^{-(t)} \geq \Delta^{+(t)}$ trivially holds, otherwise the resource exchange condition at Line 21 would not be satisfied. Finally $\Delta_{i^{+(t)}}^{+(t+1)}$ equals the value of the $j_{i^{+(t)}}^{*(t)} + 2^{\text{nd}}$ best item for player $i^{+(t)}$, $a_{i^{+(t)}, j_{i^{+(t)}}^{*(t)}+2}$, which is at least $a_{i^{+(t)}, j_{i^{+(t)}}^{*(t)}+1}$, or equivalently $\Delta^{+(t)}$. With these arguments, we reach the conclusion that $\Delta^{+(t+1)} \geq \Delta^{+(t)}$.

□

**Lemma 2.6.** Let player $i'$ be a player who has dropped an item at iteration $t'$, and suppose the resource stays constant inbetween iterations $t'$ and $t''$. If player $i'$ is the winner at iteration $t''$, i.e. i.e. picks up an item, then all of the items that have been dropped during iterations $t'+1, \dots, t''-1$ must be picked up again during iterations $t'+1, \dots, t''-1$.

*Proof.* In this analysis we will include the tie-breaker $\epsilon(i)$. For contradiction, assume otherwise; that there exists a player $i^*$ who hasn't picked up one of the items that they have dropped. Let the value of that item be $\alpha$. Notice that, since it hasn't been picked up yet,

the size of the next best item player $i^*$ will pick up at $t''$, $a_{i^*,j_{i^*}^{*(t'')}+1}$, is at most $\alpha$. Furthermore, since that item was not dropped at $t'$, the size of the biggest item player $i^*$ will drop at $t'$, $a_{i^*,j_{i^*}^{*(t')}}$, is at least $\alpha$. We know that at $t'$, $i'$ is $i^-$, i.e. $\Delta^{-(t')} = a_{i',j_{i'}^{*(t')}} + \epsilon(i') > a_{i^*,j_{i^*}^{*(t')}} + \epsilon(i^*) \geq \alpha + \epsilon(i^*)$. Similarly, at $t''$, $i'$ is $i^+$, i.e. $\Delta^{+(t'')} = a_{i',j_{i'}^{*(t'')}+1} + \epsilon(i') < a_{i^*,j_{i^*}^{*(t'')}+1} + \epsilon(i^*) \leq \alpha + \epsilon(i^*)$. By the definition of $t'$ and $t''$, we have $a_{i',j_{i'}^{*(t')}} = a_{i',j_{i'}^{*(t'')}+1}$, the item player $i'$ drops at $t'$ is the same item she picks up at $t''$. Hence, we get the following chain that yields a contradiction:

$$a_{i',j_{i'}^{*(t'')}+1} + \epsilon(i') - \epsilon(i^*) < \alpha < a_{i',j_{i'}^{*(t')}} + \epsilon(i') - \epsilon(i^*).$$

□

**Lemma 2.7.** When the CO-Lin algorithm is applied to problem (2.20), the changes in the threshold item index $j_i^*$s are monotonous for every player $i$ throughout the algorithm.

*Proof.* First, consider the increasing $j_i^*$s. By contradiction, let player $i'$ be a player who has an increase of 1 in threshold index $j_{i'}^*$ at iteration $t'$, and a later decrease of 1 at iteration $t''$, and $j_{i'}^*$ stays constant inbetween iterations $t'$ and $t''$. We have shown in Lemma 2.5 that $\Delta^{+(t')} \leq \Delta^{+(t'')}$ as $t' \leq t''$. By definition $t'$ is the iteration when $i'$ is the player with the smallest next best item and $t''$ is the iteration when $i'$ is the player with the biggest first item to drop. Therefore we know that $\Delta^{+(t')} = \Delta_{i'}^{+(t')}$ and $\Delta^{-(t'')} = \Delta_{i'}^{-(t'')}$. Furthermore, the size of the next smallest item picked at iteration $t'$ equals the size of the first item to be dropped until iteration $t''$, so $\Delta_{i'}^{+(t')} = \Delta_{i'}^{-(t)}$ for $t = t'+1, \ldots, t''$. Using these arguments, we get the chain

$$\Delta^{-(t'')} = \Delta_{i'}^{-(t'')} = \Delta_{i'}^{+(t')} = \Delta^{+(t')} \leq \Delta^{+(t'')}. \tag{2.22}$$

According to our assumption, a resource exchange occured at iteration $t''$, as player $i'$ loses an item. This is only possible if $\Delta^{-(t'')} > \Delta^{+(t'')}$ by the condition on Line 21. Therefore (2.22) yields a contradiction, i.e., there exists no such player $i'$.

Proving the monotonicity of a decreasing sequence is slightly more tedious. By contradiction, now let player $i'$ be a player who has a decrease of 1 in threshold index $j^*_{i'}$ at iteration $t'$, and a later increase of 1 at iteration $t''$, and $j^*_{i'}$ stays constant inbetween iterations $t'$ and $t''$. The increase at iteration $t''$ can either be by a swap of items, as described in Line 21, or by using up the slack, as described in Line 16. To disprove the possibility of swapping of the items, one can use a very similar argument to (2.22). The remainder of the proof focuses on disproving the possibility of using the slack.

At every iteration, resource balance is preserved; namely $S^{(t)} + \sum_{i=1}^{m} K_i^{(t)} = K$ for all $t$. Now, let's consider the resource allocations at the beginning of iteration $t'$, $\left\{K_i^{(t')}\right\}_i$, and at the beginning of $t''$, $\left\{K_i^{(t'')}\right\}_i$. By Lemma 2.6, we know that for all players $i \neq i'$, $K_i^{(t'')} \geq K_i^{(t')}$; i.e., the resource at iteration $t''$ is at least the resource at iteration $t'$ since all the items that have been dropped inbetween are picked up. Furthermore, the player $i'$ who lost at $t'$ satisfies $K_{i'}^{(t'')} = K_{i'}^{(t')} - \Delta_{i'}^{-(t')}$ by the definition of $t'$ and $t''$; and the player $i''$ who won at $t'$ satisfies $K_{i''}^{(t'')} \geq K_{i'}^{\prime(t')} + \Delta_{i''}^{+(t')}$. Thus the resource increases by at least the size of the item player $i''$ picked up at $t'$. Using these three arguments in the respective order, we get the reduction:

$$
\begin{aligned}
S^{(t'')} + \sum_{i \neq i', i''} K_i^{(t'')} + K_{i'}^{(t'')} + K_{i''}^{(t'')} &= S^{(t')} + \sum_{i \neq i', i''} K_i^{(t')} + K_{i'}^{(t')} + K_{i''}^{(t')} \\
S^{(t'')} + K_{i'}^{(t'')} + K_{i''}^{(t'')} &\leq S^{(t')} + K_{i'}^{(t')} + K_{i''}^{(t')} \\
S^{(t'')} - \Delta_{i'}^{-(t')} + K_{i''}^{(t'')} &\leq S^{(t')} + K_{i''}^{(t')} \\
S^{(t'')} - \Delta_{i'}^{-(t')} + \Delta_{i''}^{+(t')} &\leq S^{(t')}.
\end{aligned}
$$

Recall that by the definition of $i'$ and $i''$, resource exchange occurs between those players at iteration $t'$, which is only possible if (a) $\Delta_{i'}^{-(t')} > \Delta_{i''}^{+(t')}$, and (b) $S^{(t')} < \Delta_{i''}^{+(t')}$. This means that at iteration $t''$, $S^{(t'')} < \Delta_{i'}^{-(t')} = \Delta_{i'}^{+(t'')}$. Hence the slack at $t''$ is not sufficient for player $i'$ to pick up the item dropped.

□

**Theorem 2.8.** *$x^*$ is an optimal solution of problem (2.20) if it satisfies the following conditions:*

1. *For all pairs $(i_1, j_1)$ and $(i_2, j_2)$ such that $a_{i_1 j_1} < a_{i_2 j_2}$, $x^*_{i_2 j_2} = 1$ if and only if $x^*_{i_1 j_1} = 1$.*
2. *$K - \sum_{i=1}^{m} \sum_{j=1}^{N_i} a_{ij} x^*_{ij} < a_{i'j'} + K \cdot x^*_{i'j'}$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, N_i$.*

Theorem 2.8 is another way of saying that at optimality the items selected are indeed those with minimum size (Condition 1) and the remaining resource is not enough to pick up another item (Condition 2). We will prove the optimality of the CO-Lin algorithm by separately proving that it satisfies these two conditions.

**Lemma 2.8.** There does not exist two pairs $(i_1, j_1)$ and $(i_2, j_2)$ such that $a_{i_1 j_1} < a_{i_2 j_2}$ and satisfies $K^*_{i_1} < \sum_{j=1}^{j_1} a_{i_1 j}$ and $K^*_{i_2} \geq \sum_{j=1}^{j_2} a_{i_2 j}$, where $K^*_i$ is the resource allocation the CO-Lin algorithm outputs for player $i$ when applied to problem (2.21). In other words, when the CO-Lin algorithm terminates, for all pairs $(i_1, j_1)$ and $(i_2, j_2)$ such that $a_{i_1 j_1} < a_{i_2 j_2}$, either $j^*_{i_1} \geq j_1$, i.e. $x^*_{i_1 j_1} = 1$, or $j^*_{i_2} < j_2$, i.e. $x^*_{i_2 j_2} = 0$, or both.

*Proof.* By contradiction, assume that there exists such pairs $(i_1, j_1)$ and $(i_2, j_2)$. By the assumption on $K^*_{i_1}$ and $K^*_{i_2}$, $j^*_{i_1} < j_1$ (or equivalently $j^*_{i_1} + 1 \leq j_1$) and $j^*_{i_2} \geq j_2$ must hold. Since the items are in increasing order by their indices $\Delta^+_{i_1} = a_{i_1, j^*_{i_1}+1} \leq a_{i_1 j_1}$ and $\Delta^-_{i_2} = a_{i_2, j^*_{i_2}} \geq a_{i_2 j_2}$. $\Delta^+$ and $\Delta^-$ are defined such that $\Delta^+ \leq \Delta^+_{i_1}$ and $\Delta^- \geq \Delta^-_{i_2}$; and finally in the last iteration we know that $\Delta^+ \geq \Delta^-$. Using these arguments we get the chain of inequalities:

$$a_{i_1 j_1} \geq \Delta^+_{i_1} \geq \Delta^+ \geq \Delta^- \geq \Delta^-_{i_2} \geq a_{i_2 j_2}$$

contradicting our initial assumption of $a_{i_1 j_1} < a_{i_2 j_2}$. Therefore, the CO-Lin algorithm terminates by selecting items of smallest sizes.

□

**Lemma 2.9.** When the CO-Lin algorithm is applied to problem (2.21), $K-\sum_{i=1}^{m}\sum_{j=1}^{j_i^*} a_{ij} < a_{i'j'}$ holds for all $i' = 1,\ldots,m$ and $j' = j_{i'}^* + 1,\ldots,N_{i'}$ upon termination. Furthermore, this is equivalent to Condition 2 in Theorem 2.8.

*Proof.* First, recall that $K - \sum_{i=1}^{m}\sum_{j=1}^{j_i^*} a_{ij} = S$ is stored as the slack throughout the algorithm. For all pairs $(i', j')$ $i' = 1,\ldots,m$ and $j' = j_{i'}^* + 1,\ldots,N_{i'}$, in the last iteration we have $a_{i'j'} \geq a_{i'j_{i'}^*} = \Delta_{i'}^{+} \geq \Delta^{+}$. Notice that upon termination $S < \Delta^{+}$ holds, as the condition in Line 16 is not satisfied. Therefore, $K - \sum_{i=1}^{m}\sum_{j=1}^{j_i^*} a_{ij} < a_{i'j'}$ for all $i' = 1,\ldots,m$ and $j' = j_{i'}^* + 1,\ldots,N_{i'}$.

Finally, the equivalence follows from the following observations: By the definition of $j_i^*$, $\sum_{i=1}^{m}\sum_{j=1}^{N_i} a_{ij}x_{ij}^* = \sum_{i=1}^{m}\sum_{j=1}^{j_i^*} a_{ij}$. For $j' = j_{i'}^* + 1,\ldots,N_{i'}$, i.e. $x_{i'j'} = 0$, we have just shown that the condition is satisfied. For $j' = 1,\ldots,j_{i'}^*$, i.e. $x_{i'j'} = 1$, $K - \sum_{i=1}^{m}\sum_{j=1}^{j_i^*} a_{ij} < a_{i'j'} + K$ holds trivially. □

*Proof of Theorem 2.7.* Let $\{j_i'\}_{i=1}^{m}$ be the threshold indices that correspond to the resource allocation used to initialize the CO-Lin algorithm, and $\{j_i^*\}_{i=1}^{m}$ be the threshold indices when the algorithm terminates. Consider the set $S^+ = \{i : j_i^* > j_i'\}$ and a player $i \in S^+$. By the construction of our algorithm, the threshold indices increase by 1 unit only when player $i$ has the smallest next best item and is $i^+$. Lemma 2.7 implies that player $i$ is $i^+$ at exactly $j_i^* - j_i'$ iterations. The algorithm allows only 1 winner at each iteration; hence it requires $\sum_{i\in S^+} j_i^* - j_i'$ iterations for players in $i \in S^+$ to reach threshold $j_i^*$. Furthermore, at every iteration exactly one player has an increase in the threshold values. Therefore, the algorithm terminates in $\sum_{i\in S^+} j_i^* - j_i' \leq \sum_{i=1}^{m} N_i = N$ iterations.

The optimality of the output follows from Lemma 2.8, Lemma 2.9, and Theorem 2.8.

□

### 2.5.2 Communication Complexity

Communication properties of CO-Lin are identical to those of CC-Lin. Therefore all of the results discussed in Sect. 2.3.1 apply to the CO-Lin algorithm.

### 2.5.3 Generalizing the Cardinality Objective Setting by Accomodating Local Constraints

In this section we consider problems that are more general than (2.20) and have the optimality guarantee of CO-Lin. Assume that every player now has a set of local constraints and that our problem is of the form (2.23),

$$z = \max \left\{ \sum_{i=1}^{m} \mathbf{1}^T x_i : \sum_{i=1}^{m} a_i^T x_i \leq K \ , \right. \\ \left. x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \quad i = 1, \ldots, m \right\} \tag{2.23}$$

and with $\{K_i\}_{i=1}^m$ as the individual resource allocations, player $i$ has the corresponding subproblem (2.24).

$$z_i(K_i) = \max \left\{ \mathbf{1}^T x_i : a_i^T x_i \leq K_i \ , \ x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \tag{2.24}$$

Given this setting, we define the problem:

$$K_i^\theta = \min \left\{ a_i^T x_i : \mathbf{1}^T x_i \geq \theta \ , \ x_i \in X_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \tag{2.25}$$

$K_i^\theta$ is the minimum amount of resource player $i$ requires to provide a solution of cardinality at least $\theta$.

**Lemma 2.10.** For $K_i \in \left[K_i^\theta, K_i^{\theta+1}\right)$, the optimal solution of (2.24) equals $\theta$.

*Proof.* Clearly $K_i^\theta$ is a nondecreasing sequence of $\theta$, i.e. $K_i^\theta \leq K_i^{\theta+1}$. For $\theta$ such that $K_i^\theta = K_i^{\theta+1}$, the interval $\left[K_i^\theta, K_i^{\theta+1}\right)$ is empty. Hence, to prove Lemma 2.10 we may

safely assume that the $\theta$ being considered satisfies $K_i^\theta < K_i^{\theta+1}$. Notice that due to the monotonicity of the sequence, $K_i^\theta \leq K_i^{\theta'}$ holds for any $\theta' > \theta$. Let $x_i^\theta$ be the optimal solution of (2.25). $x_i^\theta \in X_i$ and $a_i^T x_i^\theta = K_i^\theta$; therefore it is a feasible solution for (2.24) for $K_i \geq K_i^\theta$ and the optimal solution of (2.24) $z_i(K_i) \geq \mathbf{1}^T x_i^\theta \geq \theta$ is lowerbounded by $\theta$. For contradiction, assume that $x_i^*$ is an optimal solution for (2.24) with $\mathbf{1}^T x_i^* = \theta' > \theta$. Since $x_i^*$ is also a feasible solution for (2.24), it satisfies $x_i^* \in X_i$ and $a_i^T x_i^* \leq K_i < K_i^{\theta+1} \leq K_i^{\theta'}$ by our assumption on $K_i$ and $K_i^{\theta+1}$. However, this contradicts our assumption on the optimal solution of (2.25) for $\theta'$; as $x_i^*$ is a feasible solution of (2.25) with $a_i^T x_i^* < K_i^{\theta'}$. Hence, no such $x_i^*$ exists, $z_i(K_i) < \theta + 1$ and we get $z_i(K_i) = \theta$. □

Similar to Sect.2.4 let us construct the following problem, in order to reduce (2.23) to (2.20)

$$\widetilde{z}_i(K_i) = \max \left\{ \mathbf{1}^T y_i \mid \widetilde{a}_i^T y_i \leq K_i,\ y_i \in \{0,1\}^{N_i} \right\} \tag{2.26}$$

where the coefficients $\widetilde{a}_{ij} = K_i^j - K_i^{j-1}$ for $j = 1, \ldots, N_i$. Notice that (2.26) is structurally equivalent to (2.21).

**Definition 2.2** (Convexity Property)**.** The problem (2.24) is defined to have the convexity property if for its corresponding problem (2.25), $K_i^{\theta+1} - K_i^\theta \leq K_i^{\theta+2} - K_i^{\theta+1}$ holds for all integer $\theta$.

**Lemma 2.11.** If problem (2.24) has convexity, then the optimal solution of (2.24) is equal to the optimal solution of (2.26) for all $K_i$ values.

*Proof.* Given an arbitrary $K_i$, let $\theta$ be such that $K_i \in \left[K_i^\theta, K_i^{\theta+1}\right)$. Lemma 2.10 implies that the optimal solution of (2.24) is $z_i(K_i) = \theta$. If (2.24) has convexity, that means that $\widetilde{a}_{ij} \leq \widetilde{a}_{i,j+1}$; and hence the optimal solution of (2.26) is the first $\theta^*$ items that satisfy $\sum_{j=1}^{\theta^*} \widetilde{a}_{ij} \leq K_i$ and $\sum_{j=1}^{\theta^*+1} \widetilde{a}_{ij} > K_i$. When we expand $\widetilde{a}_{ij} = K_i^j - K_i^{j-1}$, we get $\sum_{j=1}^{\theta^*} \widetilde{a}_{ij} = K_i^{\theta^*}$. Therefore $\widetilde{z}_i(K_i)$ is $\theta^*$ such that $K_i^{\theta^*} \leq K_i$ and $K_i^{\theta^*+1} > K_i$; and by our definition of $K_i$, $\theta^* = \theta$. □

With this reduction, we propose that player $i$, given a problem (2.24) with the convexity property, constructs problem (2.26) and runs CO-Lin on (2.26) instead. Theorem 2.9 shows that this approach has the optimality guarantee.

**Theorem 2.9.** *If (2.25) has the convexity property for all players $i = 1, \ldots, m$, then the CO-Lin algorithm applied to (2.26) outputs an optimal solution for (2.23) in $\sum_{i=1}^{m} N_i = N$ iterations.*

*Proof.* By Theorem 2.7, we know that the CO-Lin algorithm outputs an optimal resource allocation for (2.26) in $N$ iterations. Let this allocation be $\overline{K}$ and for contradiction, assume that this allocation is not optimal for (2.24). Instead, let $K^*$ be the optimal allocation for (2.24) that satisfies $\sum_{i=1}^{m} z_i(K_i^*) > \sum_{i=1}^{m} z_i(\overline{K}_i)$. By Lemma 2.11, when (2.24) has the convexity property, $z_i(K_i) = \widetilde{z}_i(K_i)$ for all $K_i$, and this would imply $\sum_{i=1}^{m} \widetilde{z}_i(K_i^*) > \sum_{i=1}^{m} \widetilde{z}_i(\overline{K}_i)$. However, the CO-Lin algorithm has the optimality guarantee for (2.26), therefore $\sum_{i=1}^{m} \widetilde{z}_i(K_i^*) \leq \sum_{i=1}^{m} \widetilde{z}_i(\overline{K}_i)$, and we get a contradiction. □

### 2.5.4 Example Problems with the Convexity Property

Revisiting Aghezzaf and Wolsey's definition in [24], for a given integral polytope $X_i$, if $conv\left(X_i \cap \left\{\sum_{j=1}^{N_i} x_{ij} = k\right\}\right)$ is integral for all nonnegative integer $k$ values, then (2.25) has the convexity property. This result directly implies that for all of the local constraints discussed in Sect. 2.4.1, CO-Lin outputs an optimal solution in at most $N$ iterations.

## 2.6 Conclusions

We proposed decentralized algorithms for solving distributed integer programming problems, where every processor is assigned equal roles and there exists no central processor that has access to every player's data. Our algorithms are designed for settings with a cardinality constraint as the binding coupling constraint, or with an arbitrary knapsack coupling constraint and a cardinality objective. The algorithms have optimality guarantees

for certain additional local constraint sets. We provide sufficient conditions for problems to guarantee optimality of the algorithms, and give examples of problems that satisfy these conditions. In the next chapter, we consider problems that don't satisfy these sufficient conditions.

# CHAPTER 3
# APPROXIMATION ALGORITHMS FOR DISTRIBUTED INTEGER PROGRAMMING PROBLEMS

## 3.1 Introduction

We assume the same resource allocation problem (2.1) in the distributed setting, where the individual players solve (2.2). The sufficient optimality condition for the decentralized CC-Lin and CC-Log algorithms in Section 2.3 only holds for problems (2.2) with the concavity property. For problems that do not have the concavity property, one option is to relax the optimality condition and consider approximations. In this chapter, we focus on heuristics with provable bounds for problems without the concavity property. We also test the performance of the CC-Lin algorithm on non-concave functions to gain some insight on the average case performance.

## 3.2 Using Concave Approximations

Consider problems (2.2) that do not have the concavity property and for which there exist concave approximation functions, an example of which is illustrated in Figure 3.1, where $z_i(\cdot)$ is the true objective function value, and $\bar{z}_i(\cdot)$ is the value of the concave approximation.

Our proposed approach in this section is to use these concave approximation functions during the decentralized algorithms CC-Lin and CC-Log, described in Section 2.3. By doing this, we give up on the optimality guarantee for our original problem; but exploit the fact that the algorithms have an optimality guarantee with respect to the concave approximation functions used. An analysis of the decentralized algorithms' accuracy when concave approximation functions are used is provided in this section. Recall that when the function is concave, CC-Lin and CC-Log output the same resource allocation, i.e. the

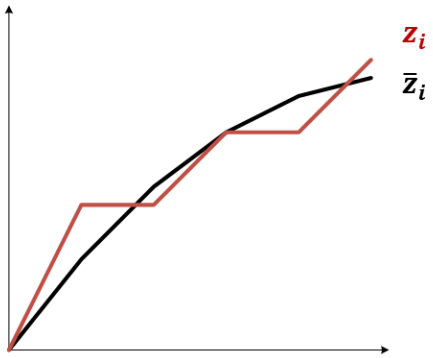



Figure 3.1: An example concave approximation

optimal allocation with respect to the concave function. Therefore, the accuracy analyses and error bounds will be the same for both of the algorithms.

### 3.2.1 Additive error bounds

For all players $i$, let $\overline{z}_i$ be an approximation of $z_i$ that has the concavity property, and satisfies

$$z_i(K_i) - \mu_i^L \leq \overline{z}_i(K_i) \leq z_i(K_i) + \mu_i^U \tag{3.1}$$

for $\mu_i^L$ and $\mu_i^U \geq 0$ and for all integer $K_i \in \{0, \ldots, K\}$. Concave overestimates of $z_i$, such as relaxations of the original problem, will have $\mu_i^L = 0$, and concave underestimates, such as heuristics for the original problem, will have $\mu_i^U = 0$. However $\overline{z}_i$ can be any arbitrary concave function that satisfies the bounds on $\mu_i^L$ and $\mu_i^U$. It is also possible to model noise in the data or errors caused by communication lags in this form.

Let $\{K_i^*\}_{i=1}^m$ be the optimal allocation of the resource and $\left\{\overline{K}_i\right\}_{i=1}^m$ be the allocation decentralized algorithms output when using $\overline{z}_i$. We define $S^+ = \left\{i \mid K_i^* > \overline{K}_i\right\}$ as the set of players who have more resource in the optimal allocation than in the output of the algorithm, $S^- = \left\{i \mid K_i^* < \overline{K}_i\right\}$ as the set of players who have less resource in the output of the algorithm than the optimal allocation. Then we have

$$\begin{aligned}
\sum_{i=1}^{m} z_i(K_i^*) - \sum_{i=1}^{m} z_i(\overline{K}_i) &= \sum_{i\in S^+} \left(z_i(K_i^*) - z_i(\overline{K}_i)\right) - \sum_{i\in S^-} \left(z_i(\overline{K}_i) - z_i(K_i^*)\right) && (3.2)\\
&\leq \sum_{i\in S^+} \left(\overline{z}_i(K_i^*) + \mu_i^L - \overline{z}_i(\overline{K}_i) + \mu_i^U\right) \\
&\quad - \sum_{i\in S^-} \left(\overline{z}_i(\overline{K}_i) - \mu_i^U - \overline{z}_i(K_i^*) - \mu_i^L\right) && (3.3)\\
&\leq \sum_{i\in S^+} \Delta_i^+(K_i^* - \overline{K}_i) - \sum_{i\in S^-} \Delta_i^-(\overline{K}_i - K_i^*) \\
&\quad + \sum_{i\in S^+\cup S^-} \mu_i^L + \mu_i^U && (3.4)\\
&= (\Delta^+ - \Delta^-) \sum_{i\in S^+} (K_i^* - \overline{K}_i) + \sum_{i\in S^+\cup S^-} \left(\mu_i^L + \mu_i^U\right) && (3.5)\\
&\leq \sum_{i\in S^+\cup S^-} \left(\mu_i^L + \mu_i^U\right) && (3.6)\\
&\leq \max_{i=1,\ldots,m} \left\{\mu_i^L + \mu_i^U\right\} \cdot \min\left\{m, K\right\}. && (3.7)
\end{aligned}$$

Since CC-Lin is applied to $\overline{z}_i$s, at the final iteration $\Delta_i^+$ and $\Delta_i^-$ are defined as $\overline{z}_i(\overline{K}_i + 1) - \overline{z}_i(\overline{K}_i)$ and $\overline{z}_i(\overline{K}_i) - \overline{z}_i(\overline{K}_i - 1)$ respectively. Furthermore, recall that $\Delta^+ = \max_{i=1,\ldots,m} \Delta_i^+$ and $\Delta^- = \min_{i=1,\ldots,m} \Delta_i^-$. Having (3.2), we replace $z_i$ with appropriate bounds of $\overline{z}_i$ to get (3.3). Knowing that $\overline{z}_i$ are concave functions of $K$, $\Delta_i^+$ is an upperbound on $\overline{z}_i(K+1) - \overline{z}_i(K)$ for $K \geq \overline{K}_i$ and $i \in S^+$, and $\Delta_i^-$ is a lowerbound on $\overline{z}_i(K) - \overline{z}_i(K-1)$ for $K \leq \overline{K}_i$ and $i \in S^-$; hence (3.4). The definition of $\Delta^+$ and $\Delta^-$ together with the fact that $\sum_{i\in S^+}(K_i^* - \overline{K}_i) = \sum_{i\in S^+}(K_i^* - \overline{K}_i) \geq 0$ imply (3.5); and $\Delta^+ \leq \Delta^-$ at termination implies (3.6). Finally, the natural upperbound on $|S_i^+ \cup S_i^-|$ gives us (3.7).

In situations where the players' problems are hard, player $i$ may use a heuristic, which yields an underestimate $z_i^H(K_i)$. In this case, comparing the true optimal, $\sum_{i=1}^m z_i(K_i^*)$, with the realized value, $z_i^H(\overline{K}_i)$ can bring additional insight. If we assume $\overline{z}_i(K_i) \leq z_i^H(K_i) + \mu_i^{HU}$ holds for the heuristic $z_i^H$, then (3.7) can be modified by similar arguments

to get

$$\sum_{i=1}^{m} z_i(K_i^*) - \sum_{i=1}^{m} z_i^H(\overline{K}_i) \leq \max_{i=1,\ldots,m} \left\{\mu_i^L + \mu_i^{HU}\right\} \cdot \min\{m, K\} \tag{3.8}$$

### 3.2.2 Multiplicative error bounds

For all players $i$, let $\overline{z}_i$ be an approximation of $z_i$ that has the concavity property, and satisfies at least one of

$$\rho_i^L z_i(K_i) \leq \overline{z}_i(K_i) \leq \rho_i^U z_i(K_i) \tag{3.9}$$

for $\rho_i^L \leq 1$ and $\rho_i^U \geq 1$. Concave overestimates of $z_i$, such as relaxations of the original problem, will have $\rho_i^L = 1$, and concave underestimates, such as heuristics for the original problem, will have $\rho_i^U = 1$. However, as stated previously, $\overline{z}_i$ can be any arbitrary concave function that satisfies the bounds on $\rho_i^L$ and $\rho_i^U$.

The analysis for $\overline{z}_i$ when it satisfies (3.9) is quite straightforward, i.e.

$$\begin{aligned}
\frac{\sum_{i=1}^m z_i(K_i^*)}{\sum_{i=1}^m z_i(\overline{K}_i)} &\leq \frac{\sum_{i=1}^m \frac{1}{\rho_i^L}\overline{z}_i(K_i^*)}{\sum_{i=1}^m \frac{1}{\rho_i^U}\overline{z}_i(\overline{K}_i)} && (3.10)\\
&\leq \frac{\frac{1}{\min_i \rho_i^L}\sum_{i=1}^m \overline{z}_i(K_i^*)}{\frac{1}{\max_i \rho_i^U}\sum_{i=1}^m \overline{z}_i(\overline{K}_i)} && (3.11)\\
&\leq \frac{\max_i \rho_i^U}{\min_i \rho_i^L}. && (3.12)
\end{aligned}$$

Since $\overline{z}_i$ has the concavity property, the output of the decentralized algorithm is optimal for $\overline{z}_i$. This means $\sum_{i=1}^m \overline{z}_i(\overline{K}_i) \geq \sum_{i=1}^m \overline{z}_i(K_i^*)$; and therefore (3.10) implies (3.12).

Similarly, if we assume there exists a heuristic $z_i^H$ for every player $i = 1, \ldots m$, that satisfies $\overline{z}_i(K_i) \leq \rho_i^{HU} z_i^H(K_i)$, thens (3.12) can be modified by similar arguments to get

$$\frac{\sum_{i=1}^m z_i(K_i^*)}{\sum_{i=1}^m z_i^H(\overline{K}_i)} \leq \frac{\max_i \rho_i^{HU}}{\min_i \rho_i^L}. \tag{3.13}$$

### 3.2.3 Example concave approximations - The greedy algorithm

Figure 3.2 is a pseudocode for the generic greedy algorithm with a cardinality constraint. Let $S_{K_i}$ be the output of the greedy algorithm for cardinality $K_i$, and define $\overline{z}_i(K_i) = f(S_{K_i})$. It is well known that the greedy algorithm has the concavity property for submodular set functions, $f$, i.e. any function $f$ that satisfies $f(S \cup \{i\}) - f(S) \geq (T \cup \{i\}) - f(T)$ where given ground set $E$, the sets $S$ and $T$ satisfy $S \subseteq T \subseteq E$, and $i \in E \backslash T$. The value of the item added is at most the value of the previous item by this submodularity inequality. The greedy algorithm, due to its simple nature, has been used as a heuristic solution method for various problems. The following papers provide the error bound of the greedy algorithm in the form of (3.9) for their respective problems, since the greedy algorithm provides a heuristic solution with $\rho_i^U = 1$. Some of these selected problems inherently have the cardinality constraint, which we require in our setting. For those that don't, we provide error bounds for the problem with the cardinality constraint included.

**Input:** Ground set of elements $E$, objective function $f : S \subseteq E \rightarrow \mathbb{R}_+$, cardinality $K$
**Output:** $S \subseteq E$ with $|S| \leq K$
1: counter = 0; $S = \emptyset$ , $N = E$
2: **while** counter $\leq K$ **do**
3: Choose $e = arg\max_{e \in N} f(S \cup \{e\}) - f(S)$
4: **if** $S \cup \{e\}$ is feasible **then**
5: $S \leftarrow S \cup \{e\}$
6: $N \leftarrow N \backslash \{e\}$
7: counter $\leftarrow$ counter + 1
8: **else**
9: $N \leftarrow N \backslash \{e\}$
10: **end if**
11: **end while**

Figure 3.2: The Greedy Algorithm with Cardinality Constraint

Nemhauser et al. [20] consider the problem of maximizing a submodular set function with respect to a cardinality constraint, defined in (3.14), and show that the ratio of the greedy solution value to the optimal solution value, i.e. $\overline{z}_i(K_i)/z_i(K_i)$ is lowerbounded by

$\rho_i^L = 1 - 1/e$.

$$\max \{z(S) : |S| \leq K,\ z(S) \text{ is submodular}\}. \tag{3.14}$$

The following problems fall into this category [20]:

- **Matroid optimization:** Given matroid $\mathcal{M} = (E, \mathcal{F})$, and weights $c_e$ for all $e \in E$, let $\mathcal{F}(E') = \{F : F \in \mathcal{F}, F \subseteq E'\}$. The set function $\nu(E')$ defined in (3.15) for all $E' \subseteq E$ is submodular.

$$\nu(E') = \max_{F \in \mathcal{F}(E')} \sum_{e \in F} c_e \tag{3.15}$$

- **Generalized transportation problem:** Given a set of sources $I$, a set of sinks $J$, and the value of assigning source $i$ to sink $k$, $c_{ik}$ for all $i \in I$ and $k \in J$. The set function $\nu(T)$ for all $T \subseteq I$ defined in (3.16) is submodular

$$\begin{aligned} \nu(T) = \max \quad & \sum_{i \in T} \sum_{k \in J} c_{ik} x_{ik} & & \\ \text{s.t.} \quad & \sum_{i \in T} x_{ik} \leq b_k & & \forall k \in J \\ & \sum_{k \in J} x_{ik} \leq 1 & & \forall i \in T \\ & x_{ik} \geq 0 & & i \in T, k \in J \end{aligned} \tag{3.16}$$

- **Boolean polynomials:** Given a ground set of elements $N$, and a real valued function $g$, the set function $\nu(S) = \sum_{T \subseteq S} g(T)$ for all $S \subseteq N$ is submodular.

Korte and Hausmann [28] investigate the problem (3.17), where $(E, \mathcal{F})$ is an indepen-

dence system, i.e. for any sets $F_1$ and $F_2$ on $E$ such that $F_1 \subseteq F_2$, $F_2 \in \mathcal{F}$ implies $F_1 \in \mathcal{F}$.

$$\max \left\{ \sum_{e \in F} c(e) : F \in \mathcal{F} \right\} \tag{3.17}$$

Their main result states that the ratio of the greedy solution value to the optimal solution value is bounded from below by $\min_{S \subseteq E} \frac{lr(S)}{ur(S)}$, where $lr(S)$ is the size of the smallest maximal independent set of $S$, and $ur(S)$ is the size of the largest maximal independent set of $S$. They further show that when $\mathcal{F}$ is an intersection of $p$ matroids, the ratio $\min_{S \subseteq E} \frac{lr(S)}{ur(S)}$ is at least $1/p$. Notice that this result applied to our problem treats the cardinality constraint in our formulation (2.2) as the $p + 1^{st}$ matroid; and the ratio becomes $1/(p+1)$. We then have the following result.

**Theorem 3.1.** *Let $(E, \mathcal{F}^i)$, $i = 1, \ldots, p$ be arbitrary matroids, $(E, \mathcal{F}^{p+1})$ be the cardinality matroid $\{F \subseteq E,\ |F| \leq K\}$ and $\mathcal{F} := \bigcap_{i=1}^{p+1} \mathcal{F}^i$. For the independence system $(E, \mathcal{F})$, $\min_{S \subseteq E} \frac{lr(S)}{ur(S)} \geq \frac{1}{p}$ holds.*

*Proof.* Let $S \subseteq E$ be any subset, and $F_1,\ F_2$ be maximal independent subsets of $S$, such that $|F_1| \leq |F_2|$. Our aim is to show that this arbitrary selection of $S$ and $(F_1, F_2)$ satisfies $|F_1|/|F_2| \geq 1/p$. Without loss of generality, we may assume that the inequality holds strictly, i.e. $|F_1| < |F_2|$, since when $|F_1| = |F_2|$ the inequality $|F_1|/|F_2| \geq 1/p$ trivially holds.

For $i = 1, \ldots, p$ and $j = 1, 2$, define $F_j^i$ to be the maximal $\mathcal{F}^i$-independent subset of $F_1 \cup F_2$ containing $F_j$. Consider an element $e \in F_2 \backslash F_1$. It is critical to observe is that $F_1 \cup \{e\}$ does not violate the cardinality constraint, i.e. $F_1 \cup \{e\} \in \mathcal{F}^{p+1}$; as $|F_1| < K$ by our initial assumption. If $e \in \bigcap_{i=1}^{p} F_1^i \backslash F_1$, then $F_1 \cup \{e\} \subseteq \bigcap_{i=1}^{p} F_1^i$. Since $F_1 \cup \{e\} \in \mathcal{F}^{p+1}$, $F_1 \cup \{e\} \in \mathcal{F}$ contradicting the maximality of $F_1$. As argued previously, $F_1 \cup \{e\} \in \mathcal{F}^{p+1}$ always holds. Hence despite the existence of cardinality matroid as the $p + 1^{st}$ matroid, each $e \in F_2 \backslash F_1$ can be an element of $F_1^i \backslash F_1$ for at most $p - 1$ of the indices $i = 1, \ldots, p$.

The remaining of the proof follows from [28].

$$\left(\sum_{i=1}^{p}|F_1^i|\right) - p|F_1| = \sum_{i=1}^{p}|F_1^i \backslash F_1| \leq (p-1)|F_2 \backslash F_1| \leq (p-1)|F_2| \tag{3.18}$$

Using (3.18) and the fact that $|F_1^i| = |F_2^i|$ for all $i = 1, \ldots, p$ since they are matroids, we derive the following:

$$\begin{aligned} |F_2| &\leq \left(\sum_{i=1}^{p}|F_2^i|\right) - p|F_2| + |F_2| \\ &= \left(\sum_{i=1}^{p}|F_1^i|\right) - (p-1)|F_2| \\ &\leq p|F_1| \end{aligned}$$

Hence the theorem holds. □

Fisher et al. [29] generalize (3.17) to the form (3.19), relaxing the linearity restriction of the objective, and generalizing it to nondecreasing submodular functions

$$\max\left\{ z(S) : S \in \bigcap_{i=1}^{p}\mathcal{F}_p,\ \mathcal{M}_p = (N, \mathcal{F}_p) \text{ are matroids, } i = 1, \ldots, p; \right. \tag{3.19}$$
$$\left. z(S) \text{ submodular and nondecreasing} \right\}.$$

They show that the ratio of the greedy solution value to the optimal solution value is bounded from below by $1/(p+1)$. Similarly, when we apply this result directly to our problem with cardinality constraint, we treat the cardinality constraint as the $p+1^{\text{st}}$ matroid and the bound becomes $1/(p+2)$. In this work, we show the following:

**Theorem 3.2.** *The ratio of the greedy solution value of the problem of maximizing a nondecreasing submodular function over the intersection of $p$ matroids and a cardinality con-*

*straint, to the optimal solution value of the same problem is at least* $\frac{1}{p+1}$.

*Proof.* Consider the following problem of maximizing a submodular set function with respect to intersection of $p+1$ matroids, where the $p+1^{\text{st}}$ matroid is a cardinality matroid.

$$\max \Bigg\{ f(S) : S \in \bigcap_{i=1}^{p} \mathcal{F}_i,\ \mathcal{M}_i = (N, \mathcal{F}_i) \text{ are matroids, } i = 1, \ldots, p; \\ |S| \leq K,\ f(S) \text{ submodular and nondecreasing} \Big\}. \tag{3.20}$$

Let $z(K)$ be the optimal solution of (3.20), and $\overline{z}_i(K_i)$ be the solution from the greedy algorithm when applied to problem (3.20). Theorem 3.2 states that $\frac{\overline{z}_i(K_i)}{z_i(K_i)} \geq \frac{1}{p+1}$.

To prove Theorem 3.2, we will modify the proof in [29]. Let $S^t$ be the first $t$ items selected in the algorithm. For any item $j \in N$ in the ground set, let $\rho_j(S) = f(S \cup \{j\}) - f(S)$ be the marginal change in the objective when $j$ is added to $S$; and $\rho_t = f(S^{t+1}) - f(S)$ be the marginal change in the objective at the $t+1^{\text{st}}$ iteration. $U^t$ will denote the set of elements considered in the first $t+1$ iterations of the greedy algorithm before the addition of the $t+1^{\text{st}}$ element; and $sp^i(S) = \{j \in N : r_i(S \cup \{j\}) = r_i(S)\}$ will denote the span of $S$ in matroid $i$ where $r_i(S)$ is the rank of set $S$ in matroid $i$.

First notice that $U^t \subseteq \cup_{i=1}^{p} sp^i(S^t)$ for $t = 0, \ldots, K-1$. Any $j \in U^t$ has either been selected, hence $j \in S^t \subseteq sp^i(S^t)$; or it has been discarded because it violated the independence of at least one of the matroids $i = 1, \ldots, p+1$. However, the cardinality of the sets $S^t$ is t, and for $t = 0, \ldots, K-1$ the cardinality of $S^t \cup \{j\}$ is at most $K$. The violated matroid cannot be the $p+1^{\text{st}}$ matroid, which is the cardinality matroid, then $j$ must have failed one of the first $p$ independence tests; meaning $j \in sp^i(S^t)$ for some $i = 1, \ldots, p$.

Let $T$ be the optimal solution and $S$ be the output of the greedy algorithm with $|S| = K' \leq K$. Further let $s_{t-1} = |T \cap (U^t - U^{t-1})|$. Our goal is to use the trivial inequality of submodular functions $f(T) \leq f(S) + \sum_{j \in T-S} \rho_j(S)$ to bound the optimal solution

$z(K) = f(T)$ and the greedy algorithm output $\overline{z}(K) = f(S)$.

The following inequality (3.21) holds since the item selection of the greedy algorithm implies $\rho_j(S) \leq \rho_{t-1}$.

$$\sum_{j \in T-S} \rho_j(S) \leq \sum_{j \in T} \rho_j(S) = \sum_{t=1}^{K'} \sum_{j \in T \cap (U^t - U^{t-1})} \rho_j(S) \leq \sum_{t=1}^{K'} \rho_{t-1} s_{t-1} \tag{3.21}$$

Next, we consider the following chain

$$\begin{aligned} \sum_{j=1}^{t} s_{j-1} &= |T \cap U^t| && (3.22) \\ &\leq \sum_{i=1}^{p} |T \cap sp^i(S^t)| && (3.23) \\ &\leq pt && (3.24) \end{aligned}$$

(3.22) holds by the definition of $s_{t-1}$; (3.23) holds by the previously shown result $U^t \subseteq \cup_{i=1}^{p} sp^i(S^t)$. Finally, since $T$ is the optimal solution, hence is independent in all $p$ matroids, $|T \cap sp^i(S^t)| \leq t$ implies (3.24).

To complete our proof, we only need Proposition 2.2 stated in [29], which is the following: if $\sum_{j=0}^{t-1} \sigma_j \leq t$ for $t = 1, \ldots, K$ and $\rho_{j-1} \geq \rho_j$, $j = 1, \ldots, K-1$ with $\rho_j, \sigma_j \geq 0$, then $\sum_{j=0}^{K-1} \rho_j \sigma_j \leq \sum_{j=0}^{K-1} \rho_j$. This proposition, along with the results (3.21) and (3.24) imply

$$f(T) \leq f(S) + \sum_{j \in T-S} \rho_j(S) \leq f(S) + p \sum_{j=0}^{K-1} \rho_j \leq (p+1) f(S)$$

Hence, our desired bound $\frac{\overline{z}_i(K_i)}{z_i(K_i)} \geq \frac{1}{p+1}$. □

### 3.2.4 Example concave approximations - Linear Relaxation

Another candidate for concave approximations is linear relaxations. Given an optimization problem with respect to a cardinality constraint where all data is integer, $z(K) = \max\left\{c^T x : Ax \leq b,\ \mathbf{1}^T x \leq K,\ x \in \{0,1\}^n\right\}$, the linear relaxation function described as $\bar{z}(K) = \max\left\{c^T x : Ax \leq b,\ \mathbf{1}^T x \leq K,\ x \in [0,1]^n\right\}$ is a concave function on integer $K$ values. Clearly $\bar{z}(K) \geq z(K)$ holds, and is not always satisfied as equality. For some problems, there are results in the literature for bounding the linear relaxation from both sides in the form of (3.9).

Bläser et al.[30] study the Max-$l$SAT problem with cardinality constraint, defined as follows: We are given a set of binary variables $x_1, \ldots, x_n$. A literal is defined to be either a variable $x_i$ or its negation $\bar{x}_i = 1 - x_i$. We are also given a set of clauses, which are disjunctions of $l$ literals. Max-$l$SAT problem with cardinality constraint, say $k$, is maximizing the number of satisfied clauses over the binary decision variables $x$, such that at most $k$ variables are set to 1. They use randomized algorithms to show that the linear relaxation has an error bound $1/\rho_i^U = 1 - \left(1 - \frac{1}{l}\right)^l - \epsilon$.

Lee et al. [31] define the weighted matchoid problem as follows: given a graph $G = (V, E)$, and a matroid $\mathcal{M}_v = (\delta(v), \mathcal{I}_v)$ for each $v \in V$, where $\delta(v) = \{e = (u, v) : e \in E\}$, find a collection of edges $F$ with maximum weight such that $F \cap \delta(v) \in \mathcal{I}_v$. Furthermore, they define the weighted matroid parity problem as, given a graph $G = (V, E)$ with the property that every vertex is incident to a single edge, i.e. $G$ is a collection of disjoint edges, and a matroid $\mathcal{M} = (V, \mathcal{I})$, find a collection of edges $F$ that forms a matching on $G$, and the end points of the edges in $F$ is independent in $\mathcal{I}$. Consider the following transformation, given a graph $G = (V, E)$, let $V'$ be the vertex set obtained by creating $|\delta(v)|$ copies of each vertex $v \in V$, i.e. $v_i, \ldots, v_{|\delta(v)|}$. For each edge $e = (u, v) \in E$, define $e' = (u_i, v_j)$, where each edge $e \in \delta(v)$ uses a different copy of $v$ while constructing $e'$. This can be interpretted as splitting each node into $|\delta(v)|$ copies where each edge in $\delta(v)$ gets its own copy of $v$ for edges in $E'$, and $G'$ becomes a collection of disjoint edges. Let the edge set

$E'$ be the collection of all such edges $e'$, where all the edges are disjoint. As discussed in [31], the weighted matchoid problem on $G = (V, E)$, with matroids $\mathcal{M}_v = (\delta(v), \mathcal{I}_v)$ for all $v \in V$ is equivalent to the weighted matroid parity on $G' = (V', E')$, with matroid $\mathcal{M}' = \cup_{v \in V} \mathcal{M}_v$.

$$z^{LP} = \max \quad \sum_{e \in E'} w_e y_e \tag{3.25}$$

$$\text{s.t.} \quad \sum_{u \in S} x_u \leq r_{\mathcal{M}'}(S) \qquad \forall S \subseteq V' \tag{3.26}$$

$$x_u = y_e \qquad u \in e, e \in E' \tag{3.27}$$

$$\sum_{u \in V'} x_u \leq 2K_i \tag{3.28}$$

$$x_u, y_e \geq 0 \qquad u \in e, e \in E' \tag{3.29}$$

The problem (3.25)-(3.27), (3.29) is a linear relaxation of the weighted matroid parity problem on $G' = (V', E')$, with matroid $\mathcal{M}' = \cup_{v \in V} \mathcal{M}_v$. The variable $y_e$ denotes whether the edge $e$ is selected. Notice that because $G'$ consists of disjoint edges, any subset of the edges yield a matching, additional constraints on $y$ are not necessary. The variable $x_u$ denotes whether node $u$ is an end point of a selected edge. By the problem definition, these nodes are required to be independent in matroid $\mathcal{M}'$, satisfied by (3.26). Clearly, $z^{LP}$ is an upperbound on the optimal solution of the weighted matchoid problem, given an upperbound of $K_i$ on the number of edges. Since for every edge $e = (u, v)$, we have $x_u = y_e = x_v$ in the corresponding matroid parity formulation, this cardinality constraint can be expressed as (3.28). Lee et al. [31] show that (3.25)-(3.27), (3.29) is a 3/2 approximation of the matchoid problem on $G$. We extend this result to show that (3.25)-(3.29) is a 3/2 approximation of matchoid problem on $G$ with a cardinality constraint of $K_i$ on the number of edges.

**Lemma 3.1.** The polyhedron defined by (3.26)-(3.29) is half integral.

*Proof.* Let $(x^*, y^*)$ be an extreme point of the polytope defined by (3.26)-(3.27), (3.29) and have $\sum_{u \in V'} x_u = K^* \leq 2K_i$. Lee et al. [31] show that $(x^*, y^*)$ can be represented as the unique solution of the system of equations (3.30)-(3.31),

$$\sum_{u \in T_{i+1} \setminus T_i} x_u = r_{\mathcal{M}'}(T_{i+1}) - r_{\mathcal{M}'}(T_i) \quad i = 0, \ldots, k-1 \tag{3.30}$$

$$x_u \geq 0 \qquad u \in V \tag{3.31}$$

$$\sum_{u \in V} x_u = K^* \tag{3.32}$$

for some set of vertices $T_0, \ldots, T_k$. Due to the structure of these sets, (3.30)-(3.31) form an integer $b$-matching polytope. In our case, if $(x^*, y^*)$ is an extreme point of the polytope defined by (3.26)-(3.29), the result in [31] directly implies that $(x^*, y^*)$ can be represented as the unique solution of (3.30)-(3.32). Hence, it suffices to show that the integer $b$-matching polytope with the cardinality constraint (3.30)-(3.32) is half integral.

□

**Proposition 3.1.** For a graph $G = (V, E)$, the $b$-matching polytope that is defined as $P = \left\{ x : \sum_{e \in \delta(v)} x_e = b_v \; \forall v \in V, x_e \geq 0 \; \forall e \in E, \sum_{e \in E} x_e = K \right\}$ is half integral.

*Proof.* Given the graph $G = (V, E)$ on which $P$ is defined, let us construct $G' = (V', E')$ in the following way: for each vertex $v \in V$, create $v', v'' \in V'$, for each edge $e = (u, v) \in E$ create $e' = (u', v'')$ and $e'' = (u'', v')$ and $e', e'' \in E'$, and $b_v = b_{v'} = b_{v''}$. Now consider the $b$-matching polyhedron on $G'$,

$$P' = \left\{ x : \sum_{e \in \delta(v)} x_e = b_v \; \forall v \in V', x_e \geq 0 \; \forall e \in E', \sum_{e \in E'} x_e = 2K \right\}. \tag{3.33}$$

Clearly $P'$ can be projected on to $P$ with the transformation $2x_e = x_{e'} + x_{e''}$. Therefore, all vertices $x^*$ of $P$ can be expressed as $x^*_e = \frac{1}{2}(\bar{x}_{e'} + \bar{x}_{e''})$ where $\bar{x}$ is a vertex of $P'$. Since $G'$ is a bipartite graph, $P'$ is an integral polytope, therefore $x^*_e \in \left\{0, \frac{1}{2}, 1\right\}$ for all $e \in E$. □

**Theorem 3.3.** *The optimal solution of the weighted matchoid problem with a cardinality constraint is at least 2/3 of the linear relaxation of the equivalent matroid parity problem with a cardinality constraint.*

*Proof.* Let $(x^*, y^*)$ be an extreme point of the polytope defined by (3.26)-(3.29). By Lemma 3.1, we know that $(x^*, y^*)$ is half integral. In order to prove Theorem 3.3, we construct a feasible solution $(\overline{x}, \overline{y})$ with weight at least 2/3 of the linear relaxation. The construction algorithm is described in [31], it starts from a half integral solution $(x^*, y^*)$ and outputs an integer solution $(\overline{x}, \overline{y})$ with weight 2/3 of the weight of $(x^*, y^*)$, while preserving cardinality, i.e. $\sum_{u \in V'} x^*_u = \sum_{u \in V'} \overline{x}_u$. Hence $(\overline{x}, \overline{y})$ is a feasible solution to the matchoid problem with weight 2/3 of the linear relaxation, meaning that $z^{LP}$ is at most 3/2 of the optimal solution.

□

### 3.3 Partial Cardinality Constraints

In this section, we consider a special case of problems that do not satisfy the concavity property. We are given problem

$$z_i(K_i) = \max \left\{ c_i^T x_i : A_i x_i \leq b_i \ , \ \alpha_i^T x_i \leq K_i \ , \ x_i \in \{0,1\}^{N_i} \right\} \tag{3.34}$$

where we assume $\alpha_{ij} \in \{0, 1\}$, and $\{x : A_i x_i \leq b_i\}$ and $\left\{x : A_i x_i \leq b_i, \mathbf{1}^T x_i = k\right\}$ are integral polytopes for all integer $k$. We consider the polytopes $\{x : A_i x_i \leq b_i\}$ such that for $\alpha_i = \mathbf{1}$, (3.34) has the concavity property. We can show by counter-examples that some problems such as the matching problem lose their concavity property when a partial cardinality constraint is included instead of full cardinality constraint. We would like to exploit this structure of (3.34) to derive concave approximation functions.

First, let us reorder the columns to reformulate (3.34) as

$$z_i(K_i) = \max \left\{ \bar{c}_i^T \bar{x}_i + \bar{\bar{c}}_i^T \bar{\bar{x}}_i : \bar{A}_i \bar{x}_i + \bar{\bar{A}}_i \bar{\bar{x}}_i \leq b_i \ , \ \mathbf{1}^T \bar{x}_i \leq K_i \ , \ (\bar{x}_i, \bar{\bar{x}}_i) \in \{0,1\}^{N_i} \right\}. \tag{3.35}$$

Define the following function:

$$z'_i(K_i) = \max \left\{ \bar{c}_i^T \bar{x}_i + \bar{\bar{c}}_i^T \bar{\bar{x}}_i : \bar{A}_i \bar{x}_i + \bar{\bar{A}}_i \bar{\bar{x}}_i \leq b_i \ , \ \mathbf{1}^T \bar{x}_i + \mathbf{1}^T \bar{\bar{x}}_i \leq K_i \ , \ (\bar{x}_i, \bar{\bar{x}}_i) \in \{0,1\}^{N_i} \right\}. \tag{3.36}$$

(3.36) is the version of (3.35) that has the full cardinality constraint, and by our assumption has the concavity property. Notice that $z'_i(K_i) \leq z_i(K_i) \leq z'_i(K_i + dim(\bar{\bar{x}}))$, as one is a restriction and the other is a relaxation of the problem (3.35). Because of the concavity property of (3.36), $z'_i(K_i) \geq \frac{K_i}{K_i + dim(\bar{\bar{x}})} z'_i(K_i + dim(\bar{\bar{x}}))$ holds. At this point, we have three candidates for the concave approximation function $\bar{z}_i$. Their respective $\rho_i^L$ and $\rho_i^U$ values for (3.9) are stated accordingly.

1. $\bar{z}_i(K_i) = z'_i(K_i) : \rho_i^L = \frac{K_i}{K_i + dim(\bar{\bar{x}})}, \rho_i^U = 1$
2. $\bar{z}_i(K_i) = z'_i(K_i + dim(\bar{\bar{x}})) : \rho_i^L = 1, \rho_i^U = \frac{K_i + dim(\bar{\bar{x}})}{K_i}$,
3. $\bar{z}_i(K_i) = \frac{1}{2}(z'_i(K_i) + z'_i(K_i + dim(\bar{\bar{x}}))) : \rho_i^L = \frac{2K_i + dim(\bar{\bar{x}})}{2(K_i + dim(\bar{\bar{x}}))}, \rho_i^U = \frac{2K_i + dim(\bar{\bar{x}})}{2K_i}$

Recall that the performance of the CC-Lin and CC-Log algorithms depend on $\rho_{min}^L$ and $\rho_{max}^U$. When $K_i = 0$, these $\rho_i^L$ and $\rho_i^U$ values turn out to be problematic and tend to blow up the ratio in (3.12). Therefore we look for a concave approximation function with an additive error bound instead. Consider the following problem:

$$\bar{z}_i(K_i) = \max \left\{ \bar{c}_i^T \bar{x}_i : \bar{A}_i \bar{x}_i \leq b_i \ , \ \mathbf{1}^T \bar{x}_i \leq K_i \ , \ \bar{x}_i \in \{0,1\}^{\bar{N}_i} \right\}. \tag{3.37}$$

(3.37) is a special case of (3.36) where $\bar{\bar{x}}_i = 0$; and therefore has the concavity property. Clearly $\bar{z}_i(K_i)$ is a lower bound on $z_i(K_i)$ as it is a restriction that sets $\bar{\bar{x}}_i = 0$. For $\bar{A}_i, \bar{\bar{A}}_i \geq 0$, $\bar{z}_i(K_i) + z_i(0)$ is an upper bound on $z_i(K_i)$ by the following argument: Let $(\bar{x}_i^*, \bar{\bar{x}}_i^*)$ be an optimal solution for (3.35) such that $z_i(K_i) = \bar{c}_i^T \bar{x}_i^* + \bar{\bar{c}}_i^T \bar{\bar{x}}_i^*$. Since $\bar{A}_i, \bar{\bar{A}}_i \geq 0$, $\bar{A}_i \bar{x}_i + \bar{\bar{A}}_i \bar{\bar{x}}_i \leq b_i$ implies $\bar{A}_i \bar{x}_i \leq b_i$ and $\bar{\bar{A}}_i \bar{\bar{x}}_i \leq b_i$. This means that $\bar{x}_i^*$ is a feasible solution to (3.37), with $\bar{z}_i(K_i) \geq \bar{c}_i^T \bar{x}_i^*$, and $(0, \bar{\bar{x}}_i^*)$ is a feasible solution to (3.35) with $z_i(0) \geq \bar{\bar{c}}_i^T \bar{\bar{x}}_i^*$. Hence we get $z_i(K_i) = \bar{c}_i^T \bar{x}_i^* + \bar{\bar{c}}_i^T \bar{\bar{x}}_i^* \leq \bar{z}_i(K_i) + z_i(0)$. Using this definition of $\bar{z}_i$ gives an additive error term $\mu_i^L + \mu_i^U = z_i(0)$ to be used in (3.7).

The problems that we mention in Section 2.4.1 satisfy $\bar{A}_i, \bar{\bar{A}}_i \geq 0$, and therefore can use $\bar{z}_i(K_i)$ defined in (3.37) as a concave approximation function with an error bound of $z_i(0)$. The strategy then becomes, given for instance a matching problem with a partial cardinality constraint, using the matching problem restricted to the edges included in the cardinality constraint, which we know has the concavity property, and incurring a possible error of at most the maximum weight matching on the graph restricted to the edges not included in the cardinality constraint. This approach may also be utilized for dealing with problems that are easy to solve with a full cardinality constraint, but become difficult with a partial cardinality constraint.

## 3.4 Knapsack Problem

In this section, we assume that every player's individual problem is a knapsack problem formulated as

$$z_i(K_i) = \max \left\{ c_i^T x_i : a_i^T x_i \leq K_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \tag{3.38}$$

Notice that the global problem is also a knapsack problem. Let us ignore the distributed

structure of the global problem and formulate it as

$$z = \max \left\{ c^T x : a^T x \leq K \ , \ x \in \{0,1\}^N \right\}. \tag{3.39}$$

The greedy algorithm is a very straightforward algorithm that can be used as a concave approximation algorithm for the knapsack problem (3.39). Without loss of generality, we assume that the items are ordered in non-increasing order of $\frac{c_i}{a_j}$ values, i.e. $\frac{c_j}{a_j} \geq \frac{c_{j+1}}{a_{j+1}}$. The greedy heuristic selects the items in this order one by one until it reaches item $k$ that satisfies $\sum_{j=1}^{k} a_j \leq K$ and $\sum_{j=1}^{k+1} a_j > K$. Despite its simplicity and concavity, the greedy algorithm performs very poorly in terms of approximation ratio. The accuracy of the greedy algorithm can be arbitrarily bad for the knapsack problem. There exists a modification of the greedy algorithm that has an approximation ratio of $1/2$. The slight modification is that at the point of termination, the algorithm compares the value of the items selected, $\sum_{j=1}^{k} c_j$ to the value of the item that is fractionally selected, $c_{k+1}$, and outputs $\max \left\{ \sum_{j=1}^{k} c_j, c_{k+1} \right\}$. This output is guaranteed to be at least $1/2$ of the optimal solution [32].

Considering this modified greedy algorithm, we propose the following modification on the CC-Lin and CC-Log algorithms. The players use the linear relaxation of the problem (3.38) as the concave approximation functions, $\bar{z}_i$, and use $\bar{z}_i$ to carry out the CC-Lin and CC-Log algorithms. Notice that the output of the linear relaxation of the knapsack problem is that the items in the greedy solution are selected, i.e. $x_j = 1$, and the next item $k+1$ is fractionally selected depending on the remaining resource.

By using tie-breaking rules we ensure that at the point the algorithm converges there is only one player with a fractionally selected item. Let $i^*$ be the player with the fractional item, and $j^*$ be the index for player $i^*$'s fractional item. The algorithm converges to a resource allocation denoted by $\bar{K}$. At the point of convergence, in order to compute the value of the greedy solution, all players except for $i^*$ broadcast $\bar{z}_i(\bar{K}_i)$, and $i^*$ broadcasts $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*})$ and $c_{i^*j^*}$. Once those pieces of information become public, players compare the value of the items selected, $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) + \sum_{i \neq i^*} \bar{z}_i(\bar{K}_i)$, and the value

of the fractional item, $c_{i^*j^*}$. If $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) + \sum_{i \neq i^*} \bar{z}_i(\bar{K}_i) \geq c_{i^*j^*}$, the algorithm outputs $\bar{K}$ as the resource allocation. Otherwise, it outputs $K_{i^*} = K$, and $K_i = 0$ for $i \neq i^*$ as the resource allocation.

**Proposition 3.2.** The modification on the CC-Lin and CC-Log algorithms explained in Section 3.4 has an approximation ratio of 1/2.

*Proof.* Let $i^*$ be the player with the fractional item after the last iteration, $j^*$ be the index for player $i^*$'s fractional item, and $\bar{K}$ be the resource allocation after the last iteration. We will consider two cases separately:

1. $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) + \sum_{i \neq i^*} \bar{z}_i(\bar{K}_i) \geq c_{i^*j^*}$:

   For $i \neq i^*$, $z_i(\bar{K}_i) = \bar{z}_i(\bar{K}_i)$ holds since the linear relaxation yields an integer solution. The same argument holds for $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) = z_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*})$. Naturally we have $z_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) \leq z_{i^*}(\bar{K}_{i^*})$, since the objective functions are nondecreasing. This means that the output of the modified greedy algorithm, $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) + \sum_{i \neq i^*} \bar{z}_i(\bar{K}_i)$ is at most the output of the modified CC-Lin algorithm, $\sum_i z_i(\bar{K}_i)$.

2. $\bar{z}_{i^*}(\bar{K}_{i^*} - a_{i^*j^*}x_{i^*j^*}) + \sum_{i \neq i^*} \bar{z}_i(\bar{K}_i) < c_{i^*j^*}$:

   Since the objective functions are nondecreasing, we have $z_{i^*}(K) \geq c_{i^*j^*}$. This means that the output of the modified greedy algorithm, $c_{i^*j^*}$ is at most the output of the modified CC-Lin algorithm, $z_{i^*}(K)$.

In either case, the true objective function value corresponding to the modified CC-Lin algorithm is at least as good as the modified greedy algorithm output. It directly follows from the modified greedy algorithm result that the ratio of the optimal solution value to the value of the modified algorithm output is at most 2.

□

## 3.5 Experimental Results

Returning back to the general problem (2.1), when the individual problems (2.2) do not have a structure like the concavity property, the algorithms can be shown to perform arbitrarily bad in terms of accuracy. Regardless of this theoretic lower bound, in this section we experiment on how they behave on average. We generate random knapsack problems and apply the CC-Lin algorithm on these non-structured problems.

Notice that regardless of the structure of the problem, the stopping condition of the CC-Lin algorithm, $\Delta^- \geq \Delta^+$, is a necessary condition for optimality. In other words, at an optimal solution, there exists no improving movements. Recall that the CC-Lin algorithm starts from an arbitrary resource allocation. If, by luck, the algorithm is initialized at the optimal solution, it terminates immediately, as there is no improving movement, and returns the optimal solution. This means that the performance of the CC-Lin algorithm strongly depends on the initial resource allocation. Along this line of thought, we propose a heuristic based on the CC-Lin algorithm that runs the CC-Lin algorithm multiple times with different initial resource allocations each time.

Another thing to consider when applying the CC-Lin algorithm to non-concave functions is that in the absence of the concavity property, we cannot guarantee that the winning player $i^+$ and the losing player $i^-$ in each iteration are distinct. Recall that the proof of Lemma 2.1 that states that $i^+$ and $i^-$ are distinct, is based on the concavity assumption, and it is easy to show examples of $i^+ = i^-$ when the problems don't have the concavity property. In order to avoid this kind of cycling, we modify the algorithm to terminate when $i^+ = i^-$. It should be noted that it is possible to enhance this approach by looking at the second best winning and losing players in the cases when $i^+ = i^-$; however, taking this approach far enough to guarantee the optimal movement will make this decision process arbitrarily difficult.

We include one final modification to our algorithm. Let player $i_1$ have resource $K_{i_1}$

which is more than she has need for, i.e. $K_{i_1} > \sum_{j=1}^{N_{i_1}} a_{i_1 j}$. In this case, the value of one less unit of resource is zero for this player, $\Delta_{i_1}^- = 0$. Now consider the remaining players, and assume that one unit of resource will not change their objective function values, and that the next jump occurs at, say, two extra units of resource. As it is, their $\Delta_i^+$ values are 0, and the algorithm does not allow the unused resource of player $i_1$ to move to other players, where it might potentially increase the objective function values in the upcoming iterations. In order to avoid this, we make the unused resource more favorable to be transferred by setting $\Delta_i^- = -\epsilon$ for players $i$ that satisfy $K_i > \sum_{j=1}^{N_i} a_{ij}$.

In our heuristic, we run the CC-Lin algorithm $m + 3$ times using the following initialization methods:

1. **ExtrPt:** We initialize the algorithm $m$ times, once for every extreme point of the polytope $\{\sum_{i=1}^m K_i = K, K_i \geq 0\}$, i.e. one player gets all the resource. This method outputs the highest value obtained from these $m$ initializations.

2. **Center:** We initialize the resource allocations to an integer point around the center of the polytope $\{\sum_{i=1}^m K_i = K, K_i \geq 0\}$, which is $\left(\frac{K}{m}\right) \mathbf{1}$. This vector is rounded such that the players with lower indices round up, the players with higher indices round down and that the sum equals $K$.

3. **Avg:** We use the average of the output resource allocations of the first $m + 1$ initializations (**ExtrPt** and **Center**) as the initial resource allocation. The average is rounded such that the players with lower indices round up, the players with higher indices round down and that the sum equals $K$.

4. **LP:** We run the CC-Lin algorithm using the LP relaxations as the concave approximation functions. We use the output of the CC-Lin algorithm on LP relaxations as the initial resource allocation. Overall, this method requires the CC-Lin algorithm to be run twice, once using the LP relaxations and once using the true objective functions.

For the following experiments, we restrict ourselves to knapsack problems (3.38) as arbitrary non-concave functions. We start with the following data set: We have $m = 10$ players and each player has $n = 10$ items. Item sizes are generated randomly such that $a_{ij} \sim U(1, 5)$, and the item values depend on the item size, $c_{ij} \sim U(a_{ij}, 2a_{ij})$. The total amount of resource $K$ is $(0.4)\sum_{i,j} a_{ij}$. We generate $T = 50$ such data sets. Tables 3.1 and 3.2 show the performance of each initialization methods on these data sets. For each data set $t$, Column 2 shows the optimal solution. Columns 3-6 show the percent deviation of the output of initialization method $A$ from the optimal solution, namely $\frac{\sum_{i=1}^{m} z_i^t(\bar{K}_i^{t,A}) - \sum_{i=1}^{m} z_i^t(K_i^{t*})}{\sum_{i=1}^{m} z_i^t(K_i^{t*})} \cdot 100$, where for initialization method $A$ and data set $t$, $\bar{K}^{t,A}$ is the output of the CC-Lin algorithm using $A$, and $K^{t*}$ is the optimal resource allocation. The last row of Table 3.2 denotes the average of these values taken over the $T$ data sets.

When the random items are distributed identically, the optimal resource allocation tends to be around the center of the feasible solution polytope. This means that the **Center** and **Avg** initialization methods terminate very quickly as their initial points are already close to the optimal solution. In order to get a better understanding of the performance of **Center** and **Avg**, we distort this structure in the following way: Every player is first assigned a number of valuable items, $\sim U(0, n)$. The item sizes are again $a_{ij} \sim U(1, 5)$, but now the value of a valuable item is $c_{ij} \sim U(5a_{ij}, 10a_{ij})$, and an low value item is $c_{ij} \sim U(1, 5a_{ij})$. The total amount of resource $K$ is $(0.4)\sum_{i,j} a_{ij}$. We generate $T = 50$ such data sets. With this modification, we increase the deviation in the random optimal resource allocation. Tables 3.3 and 3.4 show the performance of each initialization methods on these data sets. For each data set $t$, Column 2 shows the optimal solution. Columns 3-6 show the percent deviation of the output of initialization method $A$ from the optimal solution. The last row of Table 3.4 denotes the average of these values taken over the $T$ data sets.

It is clear that even with the increased deviation in the optimal resource allocations, the **LP** method for initial point selection dominates the others. After this observation, we test the robustness of the **LP** method. Namely, given a set of 100 items generated such

Table 3.1: Performance of the CC-Lin algorithm using different initialization methods for symmetric data (Data sets 1-25).

| | | Percent Deviation from Optimal | | | |
|---|---|---|---|---|---|
| Data Set | **Optimal** | **ExtrPt** | **Center** | **Avg** | **LP** |
| 1 | 234 | -6.41 | -2.56 | -4.27 | -0.43 |
| 2 | 214 | -4.21 | -3.74 | -4.21 | 0.00 |
| 3 | 236 | -4.24 | -3.81 | -1.69 | 0.00 |
| 4 | 233 | -3.00 | -3.00 | -2.15 | 0.00 |
| 5 | 230 | -3.91 | -3.04 | -2.61 | 0.00 |
| 6 | 232 | -4.74 | -2.16 | -5.17 | 0.00 |
| 7 | 234 | -7.69 | -2.99 | -4.70 | 0.00 |
| 8 | 210 | -2.86 | -2.38 | -2.86 | 0.00 |
| 9 | 220 | -4.09 | -1.36 | -3.18 | 0.00 |
| 10 | 215 | -4.19 | -3.72 | -4.65 | 0.00 |
| 11 | 231 | -4.33 | -2.16 | -3.03 | 0.00 |
| 12 | 216 | -6.02 | -1.85 | -4.17 | 0.00 |
| 13 | 250 | -5.20 | -4.80 | -6.00 | 0.00 |
| 14 | 201 | -2.99 | -3.98 | -4.48 | 0.00 |
| 15 | 234 | -6.84 | -5.13 | -4.27 | 0.00 |
| 16 | 213 | -2.35 | -1.88 | -1.88 | 0.00 |
| 17 | 221 | -4.52 | -2.71 | -5.88 | 0.00 |
| 18 | 221 | -4.98 | -2.26 | -4.07 | 0.00 |
| 19 | 225 | -4.89 | -3.11 | -2.67 | -0.44 |
| 20 | 232 | -6.47 | -3.02 | -5.17 | 0.00 |
| 21 | 228 | -5.26 | -3.07 | -2.63 | -0.44 |
| 22 | 232 | -6.03 | -1.72 | -2.59 | -0.43 |
| 23 | 209 | -4.31 | -2.87 | -2.87 | 0.00 |
| 24 | 240 | -4.17 | -3.33 | -2.08 | 0.00 |
| 25 | 223 | -4.93 | -4.93 | -2.69 | 0.00 |

that $a_{ij} \sim U(1,5)$ and $c_{ij} \sim U(1, 10a_{ij})$, we distribute them among $m = 2, 5, 10, 20, 50$ players, and solve the resulting distributed problem using the **LP** initialization method. Table 3.5 shows the performance of each $m$ value for each data set. For each data set $t$, Column 2 shows the optimal solution. Columns 3-7 show the percent deviation of the output of $m$ value from the optimal solution. The last row of Table 3.5 denotes the average of these values taken over the $T$ data sets.

The results are not surprising. Let $x^{LP}$ be the optimal solution of the linear relaxation,

Table 3.2: Performance of the CC-Lin algorithm using different initialization methods for symmetric data (Data sets 26-50).

| | | Percent Deviation from Optimal | | | |
|---|---|---|---|---|---|
| Data Set | **Optimal** | **ExtrPt** | **Center** | **Avg** | **LP** |
| 26 | 224 | -5.80 | -4.46 | -3.57 | 0.00 |
| 27 | 215 | -4.19 | -2.33 | -4.19 | -0.47 |
| 28 | 230 | -9.13 | -3.91 | -6.52 | 0.00 |
| 29 | 224 | -3.57 | -3.57 | -1.79 | -0.45 |
| 30 | 225 | -5.78 | -5.78 | -4.44 | -0.44 |
| 31 | 217 | -6.45 | -5.99 | -6.91 | 0.00 |
| 32 | 217 | -2.76 | -2.76 | -1.84 | 0.00 |
| 33 | 210 | -3.81 | -2.86 | -3.33 | 0.00 |
| 34 | 235 | -5.11 | -4.68 | -2.55 | 0.00 |
| 35 | 228 | -7.02 | -5.70 | -5.26 | -0.44 |
| 36 | 217 | -5.53 | -3.23 | -2.30 | 0.00 |
| 37 | 205 | -4.39 | -2.93 | -1.95 | 0.00 |
| 38 | 205 | -3.90 | -1.95 | -1.46 | 0.00 |
| 39 | 217 | -4.61 | -2.76 | -2.76 | -0.46 |
| 40 | 226 | -4.42 | -2.65 | -1.33 | 0.00 |
| 41 | 214 | -2.34 | -4.67 | -1.87 | 0.00 |
| 42 | 217 | -5.53 | -2.30 | -4.15 | -0.46 |
| 43 | 232 | -4.31 | -1.72 | -1.29 | 0.00 |
| 44 | 225 | -4.00 | -3.56 | -0.89 | 0.00 |
| 45 | 227 | -3.96 | -1.76 | -3.52 | 0.00 |
| 46 | 229 | -3.06 | -5.68 | -2.62 | 0.00 |
| 47 | 216 | -3.70 | -3.70 | -1.85 | 0.00 |
| 48 | 217 | -5.07 | -4.15 | -1.84 | 0.00 |
| 49 | 214 | -3.74 | -2.34 | -0.93 | 0.00 |
| 50 | 219 | -3.20 | -2.28 | -0.91 | 0.00 |
| **Average** | **222.78** | **-4.68** | **-3.27** | **-3.20** | **-0.09** |

and $x^*$ be the true optimal solution. Even though the vectors $x^{LP}$ and $x^*$ are not necessarily close to each other, as the items are distributed among more players, i.e. as $m$ increases, for each player the difference between the optimal resource allocation $\sum_j a_{ij}x^*_{ij}$ and the linear relaxation resource allocation $\sum_j a_{ij}x^{LP}_{ij}$ increases. This difference causes the decrease in the performance of the algorithm. Overall, we can conclude that the **LP** initialization method outperforms the other three methods, due to its robustness. The average percent deviation from the optimal solution is less than 1 % for the **LP** initialization method.

Table 3.3: Performance of the CC-Lin algorithm using different initialization methods for asymmetric data (Data sets 1-25).

| | | Percent Deviation from Optimal | | | |
|---|---|---|---|---|---|
| Data Set | **Optimal** | **ExtrPt** | **Center** | **Avg** | **LP** |
| 1 | 1014 | -7.40 | -7.40 | -5.72 | -0.10 |
| 2 | 986 | -7.20 | -5.17 | -3.45 | 0.00 |
| 3 | 918 | -14.81 | -1.31 | -8.50 | 0.00 |
| 4 | 1068 | -12.36 | -11.52 | -23.22 | -0.19 |
| 5 | 934 | -1.82 | -7.71 | -1.07 | 0.00 |
| 6 | 825 | -1.70 | -11.88 | -3.03 | 0.00 |
| 7 | 1060 | -8.30 | -2.08 | -5.28 | -0.38 |
| 8 | 887 | -0.68 | -3.83 | -0.45 | -0.23 |
| 9 | 969 | -16.72 | -4.64 | -10.32 | 0.00 |
| 10 | 1003 | -0.90 | -5.98 | -1.30 | 0.00 |
| 11 | 978 | -2.97 | -2.66 | -2.86 | -0.20 |
| 12 | 1021 | -9.11 | -5.48 | -5.88 | -0.20 |
| 13 | 1028 | -7.20 | -3.99 | -5.45 | 0.00 |
| 14 | 1171 | -4.78 | -1.79 | -3.50 | -0.09 |
| 15 | 1014 | -4.24 | -2.66 | -6.80 | 0.00 |
| 16 | 1005 | -6.17 | -7.26 | -5.67 | 0.00 |
| 17 | 945 | -4.23 | -3.39 | -4.55 | -0.11 |
| 18 | 1020 | -8.14 | -11.08 | -8.63 | -0.49 |
| 19 | 886 | -6.55 | -5.30 | -6.32 | -0.11 |
| 20 | 982 | -2.55 | -2.44 | -0.81 | -0.20 |
| 21 | 1065 | -6.10 | -4.69 | -7.32 | 0.00 |
| 22 | 1092 | -3.66 | -3.30 | -0.92 | 0.00 |
| 23 | 972 | -6.89 | -3.81 | -8.44 | 0.00 |
| 24 | 957 | -2.82 | -9.30 | -1.15 | -0.31 |
| 25 | 960 | -5.52 | -7.92 | -5.42 | 0.00 |

## 3.6 Conclusions

In this chapter we proposed decentralized approximation algorithms based on the CC-Lin and CC-Log algorithms described in Chapter 2, to be used on problems that don't have the concavity property. We provided lower bounds on the performance of these approximation algorithms when the problems without the concavity property have certain known structures. We also tested the performance of the approximation heuristic in the absence of additional structures. Despite the weak lower bound on the worst-case accuracy, the

Table 3.4: Performance of the CC-Lin algorithm using different initialization methods for asymmetric data (Data sets 26-50).

| | | Percent Deviation from Optimal | | | |
|---|---|---|---|---|---|
| Data Set | **Optimal** | **ExtrPt** | **Center** | **Avg** | **LP** |
| 26 | 791 | -5.06 | -5.44 | -3.16 | 0.00 |
| 27 | 994 | -5.63 | -8.45 | -6.84 | -0.30 |
| 28 | 933 | -13.50 | -6.00 | -7.93 | -0.11 |
| 29 | 1053 | -3.04 | -3.51 | -1.52 | 0.00 |
| 30 | 890 | -1.80 | -4.38 | -1.46 | -0.11 |
| 31 | 1065 | -8.08 | -1.97 | -5.73 | -0.38 |
| 32 | 980 | -5.10 | -2.76 | -0.71 | -0.10 |
| 33 | 917 | -5.67 | -3.60 | -5.89 | -0.44 |
| 34 | 892 | -6.39 | -8.74 | -5.38 | -0.11 |
| 35 | 874 | -12.47 | -3.89 | -5.84 | 0.00 |
| 36 | 983 | -6.51 | -4.88 | -5.09 | 0.00 |
| 37 | 913 | -2.74 | -3.50 | -3.50 | 0.00 |
| 38 | 768 | -2.99 | -4.56 | -2.34 | 0.00 |
| 39 | 1019 | -8.44 | -5.40 | -5.69 | 0.00 |
| 40 | 969 | -4.64 | -3.92 | -2.79 | -0.21 |
| 41 | 947 | -5.60 | -2.11 | -2.75 | 0.00 |
| 42 | 964 | -18.46 | -5.50 | -14.21 | -0.21 |
| 43 | 988 | -7.39 | -7.79 | -7.39 | 0.00 |
| 44 | 869 | -10.24 | -7.36 | -6.44 | 0.00 |
| 45 | 1072 | -5.32 | -5.22 | -8.12 | 0.00 |
| 46 | 942 | -4.88 | -8.81 | -3.50 | 0.00 |
| 47 | 1023 | -7.14 | -2.25 | -3.03 | 0.00 |
| 48 | 902 | -8.98 | -8.87 | -6.76 | -0.22 |
| 49 | 970 | -5.98 | -7.01 | -8.97 | 0.00 |
| 50 | 901 | -8.77 | -4.88 | -8.21 | 0.00 |
| **Average** | **967.58** | **-6.55** | **-5.35** | **-5.39** | **-0.10** |

CC-Lin algorithm using the **LP** initialization method is within 1% error margin from the optimal solution on average.

Table 3.5: Performance of the CC-Lin algorithm using the **LP** initialization method for different $m$ values.

| | | Percent Deviation from Optimal | | | | |
|---|---|---|---|---|---|---|
| Data Set | **Optimal** | $m = 2$ | $m = 5$ | $m = 10$ | $m = 20$ | $m = 50$ |
| 1 | 1030 | 0.00 | -0.29 | -0.78 | -0.87 | -0.87 |
| 2 | 1028 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 987 | 0.00 | 0.00 | 0.00 | -0.20 | -0.61 |
| 4 | 1038 | 0.00 | -0.19 | -0.58 | -0.58 | -0.58 |
| 5 | 1056 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 6 | 1020 | 0.00 | 0.00 | -0.88 | -1.57 | -1.57 |
| 7 | 948 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 8 | 1008 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 9 | 1012 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 10 | 1043 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 11 | 981 | 0.00 | 0.00 | 0.00 | 0.00 | -2.04 |
| 12 | 1007 | 0.00 | 0.00 | 0.00 | 0.00 | -1.79 |
| 13 | 991 | -0.10 | 0.00 | 0.00 | 0.00 | 0.00 |
| 14 | 1003 | -0.20 | -0.20 | -0.50 | -0.20 | -0.20 |
| 15 | 881 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 16 | 1066 | 0.00 | 0.00 | 0.00 | -0.47 | 0.00 |
| 17 | 1063 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 18 | 1035 | 0.00 | 0.00 | 0.00 | 0.00 | -0.58 |
| 19 | 945 | 0.00 | 0.00 | 0.00 | 0.00 | -0.85 |
| 20 | 986 | -0.10 | -0.20 | -0.51 | -0.61 | 0.00 |
| 21 | 955 | -0.10 | -0.31 | -0.31 | -0.10 | -0.10 |
| 22 | 1008 | 0.00 | -0.10 | -0.30 | 0.00 | 0.00 |
| 23 | 968 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 24 | 920 | 0.00 | 0.00 | -0.43 | -0.54 | 0.00 |
| 25 | 1027 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 26 | 939 | 0.00 | 0.00 | -0.43 | -0.53 | -0.53 |
| 27 | 927 | 0.00 | -0.11 | -0.11 | -0.11 | -1.40 |
| 28 | 1030 | -0.10 | -0.19 | -0.19 | -0.39 | -0.10 |
| 29 | 944 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 30 | 876 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Average** | **990.73** | **-0.02** | **-0.05** | **-0.17** | **-0.21** | **-0.37** |

# CHAPTER 4

# DECENTRALIZED ONLINE INTEGER PROGRAMMING

## 4.1 Introduction

In this chapter we will assume the problem structure (2.1) described in Chapter 2 in an online setting. Namely, we consider a set of collaborative agents that need to coordinate their actions so as to socially optimize the sum of their objectives while satisfying a common resource constraint. It is a repetitive game, where the objective functions of the players are unknown to them a priori and are revealed in an online manner over time. Given a resource allocation, each player's action is determined by solving an integer program. Due to privacy issues, players want to share limited information while solving this problem in a decentralized way. A cardinality resource constraint links all player actions. The resulting problem is an online optimization problem to optimally allocate the resource among the players prior to observing the item values.

Such problems arise in digital platforms. One example problem is advertisement allocations on search pages, where the players are the companies wanting to purchase ad slots, the common resource is the number of ad slots on the page, and the value of the ad slots change over time. Another example is allocating IP addresses to local authorities, where the demand, i.e. the value, of the resource changes over time.

In the most general form, our problem can be formulated as follows:

$$z^t = \max \left\{ \sum_{i=1}^{m} z_i^t(K_i) : \sum_{i=1}^{m} K_i = K \right\}. \tag{4.1}$$

The set $i = 1, \ldots, m$ corresponds to players. The global objective is the sum of individual players' objective functions, $z_i^t(K_i)$. Each player $i$ has their individual problem (4.2), coupled with a cardinality constraint. The value $K$ can be interpreted as a common

resource the players share. For every $t$, we want to find an optimal resource allocation, $K^{*t}$, such that $\sum_{i=1}^{m} z_i^t(K_i^{*t}) = z^t$. For the sake of simplicity, we start with

$$z_i^t(K_i) = \max \left\{ (c_i^t)^T x_i : \mathbf{1}^T x_i \leq K_i \ , \ x_i \in \{0,1\}^{N_i} \right\}. \tag{4.2}$$

as the item selection problem. We further show in Section 4.7, that the results in this chapter hold for all problems satisfying the *concavity property*.

At every iteration the values of the items change, and the online problem is allocating the resource to players prior to observing the item values. Our goal is to derive decentralized online optimization algorithms with provably good performances, that take $\left\{z_i^1, \ldots, z_i^{t-1}\right\}_{i=1}^m$ as input, and output a resource allocation $\{K_i^t\}_{i=1}^m$.

## 4.2 Literature Survey

Bubeck [17] and Hazan [16] provide two extensive pieces of work that cover a wide range of topics in online convex optimization. There exist numerous efficient online convex optimization algorithms in the literature with good error bounds, such as *online gradient descent* and its generalization *online mirror descent*. These convex optimization algorithms assume a smooth transition over the decision variables throughout the iterations. However, the decision variables in our problem (4.1), i.e. the resource allocations, are in the discrete domain. Therefore we cannot make use of the online convex optimization algorithms directly to solve our problem.

The most elementary version of our problem is described by Koolen et al. [18]. The algorithm they propose works on discrete online optimization problems where the decision variables are binary, i.e. $y^t \in \{0,1\}^m$, and the individual objective functions are linear, i.e. $z_i^t(y_i) = l_i^t y_i$ for every player $i = 1, \ldots, m$. Their problem is more general in the sense that it admits any subset of the unit hypercube to be the set of feasible decisions. The algorithm that they propose has an a smallest possible error bound for this class of problems.

Audibert et al. [19] consider the same problem setting, and generalize the algorithm in [18] to online stochastic mirror descent. They further provide analysis for when instead of observing the whole $l^t$ vector at the end of each iteration, which is known as the full-information setting, only the components of the $l^t$ vector for which $y_i^t = 1$ are revealed, referred to as the semi-bandit feedback setting, or only the value of the decision, i.e. $(l^t)^T y^t$, is revealed, referred to as the bandit feedback setting.

Liu and Zhao [33] solve the multi-armed bandit problem with multiple players in a distributed setting. This problem is defined as follows: There are $j = 1, \ldots, N$ arms and each arm $j$ returns a reward $\theta_j^t$ at each iteration $t$. Each player $i = 1, \ldots, m$ picks an arm to play at every iteration $t$, and receives the respective reward. When multiple players select the same arm, the reward $\theta_j^t$ is shared among them. The main difference between this problem and ours is that we assume each player's decision set is disjoint.

Braun and Pokutta [34] consider the problem where the decisions $x^t \in A \subseteq \mathbf{R}^n$ from a finite set yield a linear loss $(L^t)^T x^t$. They assume the linear loss functions $L^t$ are output by adaptive adversaries, and provide an efficient algorithm against them.

Our work is along the lines of [18] and [19]. Our contribution in this paper is to generalize the binary problem to integer programming problems, and the linear objective function to step functions. Furthermore, we focus on the decentralization aspect of the problem and provide a decentralized protocol that shares a linear number of messages in $m$, i.e. the number of players, at every iteration.

### 4.3 Deterministic Algorithms

Consider an online optimization problem that is solved for $T$ iterations. A generally accepted measure for the performance of an online optimization algorithm $\mathcal{A}$ is the regret function defined as

$$R_{\mathcal{A}}(T) = \sup_{z_i^t \in \mathcal{F}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^t) \right\} \tag{4.3}$$

where in our case $\mathcal{F}$ is the set of objective functions $z_i^t$ that can be expressed as (4.2), and $K^* = \{K_i^*\}_{i=1}^m$ is the best fixed decision, i.e.

$$K^* = \arg\max \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i) \ : \ \sum_{i=1}^{m} K_i = K, \ \ K_i \in \mathbf{Z}_+ \ \forall i = 1, \ldots, m \right\}. \qquad (4.4)$$

In other words, regret measures the maximum difference between the value of the best static decision, $\sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*)$, and the value of the algorithm output, $\sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^t)$ over all possible objective functions in order to consider the worst-case scenario. Notice that the comparison is made with the best static decision, not the true optimal decision that is allowed to have different values at each iteration. The reason behind that is usually explained as fairness. Namely, the optimal solution with full information revealed prior to decision making is too advantageous compared to the decisions made prior to observing the data. Because $K^*$ is not the optimal resource allocation independently for all $t$, the regret $R_{\mathcal{A}}(T)$ is not necessarily a nonnegative value.

An algorithm is accepted to have a good performance if the regret is bounded from above by a sublinear function of $T$, i.e. $R_{\mathcal{A}}(T)$ is $o(T)$ [16]. There are many online convex optimization algorithms in the literature that satisfy this regret bound. For instance the online gradient descent algorithm, and even its generalization the online mirror descent algorithm have upper bounds of $O(\sqrt{T})$ on regret in online convex optimization problems. If the problem satisfies some additional conditions, such as being $\alpha$-strongly concave, we can get an upper bound of $O(\log T)$ on regret for the online gradient descent algorithm [16]. We show in Theorem 4.1 that for our problem, the desired bound of $o(T)$ on $R_{\mathcal{A}}(T)$ cannot be obtained by a deterministic online algorithm.

**Theorem 4.1.** *For every deterministic online algorithm $\mathcal{A}$, there exists a sequence of objective functions $\left\{z_i^1, \ldots, z_i^T\right\}_{i=1}^m$ in the family $\mathcal{F}$, that guarantees $R_{\mathcal{A}}(T)$ is $\Omega(T)$.*

*Proof.* To prove this theorem, we consider a special case of our problem with 2 players,

$m = 2$, and a single unit of resource, $K = 1$. With this assumption, our problem reduces to the experts problem described in [16]. Namely given an objective function vector $c^t$ of dimension $m$, the problem is proposing a decision vector $x^t = \{0,1\}^m$ with $|x^t| = 1$ to maximize $(c^t)^T x^t$. Given an arbitrary deterministic online algorithm $\mathcal{A}$, we will construct the adversary functions $\left\{z_i^1, \ldots, z_i^T\right\}_{i=1}^m$, i.e. the nonnegative item values $\left\{c_i^1, \ldots, c_i^T\right\}_{i=1}^m$ that will give us the $\Omega(T)$ lower bound. Notice that since $K = 1$, $z_i^t(K_i^t) = c_i^t K_i^t$.

Let algorithm $\mathcal{A}$ output resource allocation $(K_1^t, K_2^t) \in \{(1,0),(0,1)\}$ at iteration $t$. After fixing the decision, item values are observed, and the adversary item values are such that $c_i^t = 1 - K_i^t$. In other words, the valuable item goes to the player with no resource and the player with the resource receives an item of no value. These adversary item values guarantee that $z_i^t(K_i^t) = 0$. As for the best fixed decision $(K_1^*, K_2^*) \in \{(1,0),(0,1)\}$, notice that the value of the best decision is at least $T/2$ since the values of the two feasible decisions add up to $T$. With this, we reach the following conclusion:

$$R_{\mathcal{A}}(T) \geq \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^t) \geq T/2 - 0 \geq \Omega(T). \tag{4.5}$$

□

## 4.4 Randomized Algorithms

Theorem 4.1 implies that we cannot achieve the desired bound $R_{\mathcal{A}}(T) \leq o(T)$ by a deterministic online algorithm. Hence, we further look into randomized algorithms. Given the objective functions $\left\{z_i^1, \ldots, z_i^{t-1}\right\}_{i=1}^m$ as input, at iteration $t$ a randomized algorithm $\mathcal{A}$ outputs resource allocation $K^t \in \left\{\bar{K}^1, \ldots, \bar{K}^r\right\}$ with probabilities $\lambda_1^t, \ldots, \lambda_r^t$ respectively. Let this randomization be such that the expectation $\mathbf{E}[K^t]$ equals the vector $\omega^t$, i.e. the probabilities $\lambda_1^t, \ldots, \lambda_r^t$ satisfy $\sum_{j=1}^r \lambda_j^t \bar{K}^j = \omega^t$. The randomized algorithm updates the expectation vector $\omega^t$ at every iteration.

The performance of a randomized online optimization algorithm is determined by the

expected regret function defined as

$$R_{\mathcal{A}}^{E}(T) = \sup_{z_i^t \in \mathcal{F}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} \mathbf{E}\left[ z_i^t(K_i^t) \right] \right\} \tag{4.6}$$

where $K^*$ is the best static decision described in (4.4).

The structure of the problem in (4.2) is such that $z_i^t(K_i)$ equals the sum of the highest $K_i$ elements in the vector $c_i^t$ for $K_i \in \mathbf{Z}_+$, and $z_i^t(K_i) = z_i^t(K_i + \epsilon)$ for $K_i \in \mathbf{Z}_+$ and $\epsilon \in [0, 1)$. It is a step-function as illustrated in Figure 4.1. Consider the upper envelope of $z_i^t$, denoted by $\tilde{z}_i^t$, in other words the linear relaxation of the problem (4.2). By the structure of our problem, $\tilde{z}_i^t(K_i) = z_i^t(K_i)$ for all $K_i \in \mathbf{Z}_+$.

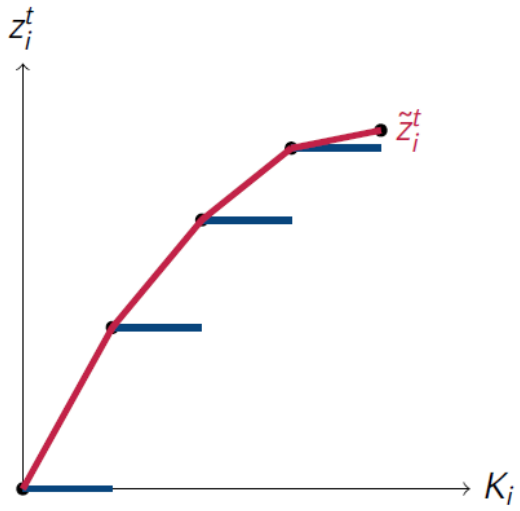



Figure 4.1: Functions in $\mathcal{F}$ and their upper envelopes

Let us construct the following online convex optimization problem over the continuous concave functions $\tilde{z}_i^t$ to be maximized

$$\tilde{R}_{\mathcal{A}}(T) = \sup_{\tilde{z}_i^t \in \tilde{\mathcal{F}}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(\tilde{K}_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(\omega_i^t) \right\} \tag{4.7}$$

where the feasible decisions are $\omega^t \in \left\{ \omega \in \mathbf{R}_+^m \,,\ \sum_{i=1}^m \omega_i = K \right\}$, $\tilde{\mathcal{F}}$ is the family of functions that can be described as the upper envelope of functions in $\mathcal{F}$, and $\tilde{K}^* = \left\{ \tilde{K}_i^* \right\}_{i=1}^m$

is the best fixed decision in the continuous space, i.e.

$$\tilde{K}^* = \arg\max \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(K_i) \ : \ \sum_{i=1}^{m} K_i = K, \ \ K_i \geq 0 \ \forall i = 1, \ldots, m \right\}. \tag{4.8}$$

**Theorem 4.2.** *For a function $z_i^t \in \mathcal{F}$, and its upper envelope $\tilde{z}_i^t$, if $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$ holds, then for any online convex optimization algorithm $\mathcal{A}$, $R_{\mathcal{A}}^E(T) \leq \tilde{R}_{\mathcal{A}}(T)$.*

*Proof.* For any fixed function $z_i^t \in \mathcal{F}$ and its upper envelope $\tilde{z}_i^t$, the following statements are true:

- $z_i^t(K_i^*) \leq \tilde{z}_i^t(K_i^*)$ ($\tilde{z}_i^t$ is the upper envelope of $z_i^t$)
- $\sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(K_i^*) \leq \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(\tilde{K}_i^*)$ ($\tilde{K}^*$ is the maximizer and $K^*$ is a feasible solution of (4.8) )
- $\tilde{z}_i^t(\omega_i^t) = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$ (the definition of $\mathbf{E}\left[K_i^t\right]$)
- $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$ (our assumption)

Hence we get the following chain:

$$\sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} \mathbf{E}\left[z_i^t(K_i^t)\right] \leq \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(\tilde{K}_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} \tilde{z}_i^t(\omega_i^t). \tag{4.9}$$

□

The importance of Theorem 4.2 is that as long as the algorithm satisfies $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$, we can utilize any online convex optimization algorithm on the linear relaxations $\left\{\tilde{z}_i^1, \ldots, \tilde{z}_i^T\right\}_{i=1}^m$ over the expectation vectors $\omega^t$, and the upper bound on the regret of the convex optimization algorithm holds as an upper bound on the expected regret $R_{\mathcal{A}}^E(T)$.

## 4.5 A Decentralized Randomized Online Algorithm

The main framework of our algorithm is the *Online Stochastic Mirror Descent* in [19] using the update formulas in [18]. We describe this framework in Figure 4.2. At every iteration, the expectation vector $\omega^t$ is updated in two steps. The intermediary point $\hat{\omega}^{t+1}$ is computed in Line 3 by optimizing the tradeoff between the relative entropy with respect to the previous iterate $\omega^t$, $\sum_{i=1}^m \left(\omega_i \ln \frac{\omega_i}{\omega_i^t} + \omega_i^t - \omega_i\right)$, and a linear approximation of the objective function, $\sum_{i=1}^m (\omega_i \nabla \tilde{z}_i^t(\omega_i^t))$. This is an unconstrained optimization problem and has a closed form solution, $\hat{\omega}_i^{t+1} = \omega_i^t \, e^{\eta \nabla \tilde{z}_i^t(\omega_i^t)}$. $\omega^t$ is a feasible solution, i.e. $\sum_{i=1}^m \omega_i^t = K$ and $\omega^t \geq 0$. With this update formula, $\hat{\omega}^{t+1}$ is not feasible, i.e. $\sum_{i=1}^m \hat{\omega}_i^{t+1} > K$, unless $\nabla \tilde{z}_i^t(\omega_i^t) = 0$ for all $i = 1, \ldots, m$. In Line 4, $\hat{\omega}^{t+1}$ is then projected back to the feasible solution polytope by minimizing the relative entropy with respect to $\hat{\omega}^{t+1}$. This problem also has a closed form solution, $\omega_i^{t+1} = \frac{\hat{\omega}_i^{t+1}}{\sum_{j=1}^m \hat{\omega}_j^{t+1}}$.

At iteration $t$:

1: Pick $K^t$ randomly such that $\omega^t = \mathbf{E}\left[K_i^t\right]$ and $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$.

2: Observe the new objective function $\{z_i^t(\cdot)\}_{i=1}^m$

3: $\hat{\omega}^{t+1} = \arg\min_{\omega \in \mathbf{R}^m} \left\{ \sum_{i=1}^m \left(\omega_i \ln \frac{\omega_i}{\omega_i^t} + \omega_i^t - \omega_i\right) - \eta \sum_{i=1}^m (\omega_i \nabla \tilde{z}_i^t(\omega_i^t)) \right\}$
$\Rightarrow \; \hat{\omega}_i^{t+1} = \omega_i^t \, e^{\eta \nabla \tilde{z}_i^t(\omega_i^t)}$

4: $\omega^{t+1} = \arg\min_{\omega} \left\{ \sum_{i=1}^m \left(\omega_i \ln \frac{\omega_i}{\hat{\omega}_i^{t+1}} + \hat{\omega}_i^{t+1} - \omega_i\right) : \sum_{i=1}^m \omega_i = K, \omega \geq 0 \right\}$
$\Rightarrow \; \omega_i^{t+1} = \frac{\hat{\omega}_i^{t+1}}{\sum_{j=1}^m \hat{\omega}_j^{t+1}}$

Figure 4.2: The centralized framework.

The structure of the relative entropy function is very convenient for decentralization. As stated in Line 3 of Figure 4.2, the updates on $\hat{\omega}^t$ are separable over the players. The updates on $\omega^t$, on the other hand, require the sum $\sum_{j=1}^m \hat{\omega}_j^{t+1}$. Sharing this sum, denoted by the ***ShareSum*** subroutine, can be achieved using at most $O(m)$ messages by protocols such as the *Secure Sum Protocol* described in [35]. Furthermore, in order to coordinate the selection of $K^t$, the updates on $\omega^{t+1}$ need to be shared among all players. Communi-

cation complexity of this protocol, *multinode broadcast*, where everyone sends everyone a message is $\Theta(m)$ as stated in [22]. With this decentralization we get the decentralized algorithm described in Figure 4.3.

Player $i$'s routine at iteration $t$:

1: $K_i^t = \textbf{\textit{Pick}} K_i^t \textbf{\textit{Randomly( )}}$
2: Observe the new objective function $z_i^t(\cdot)$
3: $\hat{\omega}_i^{t+1} = \omega_i^t \, e^{\eta \nabla \tilde{z}_i^t(\omega_i^t)}$
4: $\sum_{j=1}^m \hat{\omega}_j^{t+1} = \textbf{\textit{ShareSum}}(\hat{\omega}_i^{t+1})$
5: $\omega_i^{t+1} = \frac{\hat{\omega}_i^{t+1}}{\sum_{j=1}^m \hat{\omega}_j^{t+1}}$
6: $\omega^{t+1} = \textbf{\textit{ShareOmega}}\left(\omega_i^{t+1}\right)$

Figure 4.3: The decentralized algorithm.

The main challenge while decentralizing the algorithm in Figure 4.2 to get the algorithm in Figure 4.3 is coordination while randomly picking the resource allocation vector $K^t$. The subroutine that we propose for the random selection of $K^t$ is along the line of the proof of the Caratheodory theorem explained in [36]. First of all, we restrict ourselves to the set of feasible resource allocations defined as

$$\mathcal{K}^{\omega^t} = \left\{ \bar{K} \ : \ \bar{K} \in \mathbf{Z}_+^m \ , \ \sum_{i=1}^m \bar{K}_i = K \ , \ \bar{K}_i \in \left\{ \lfloor \omega_i^t \rfloor, \lceil \omega_i^t \rceil \right\} \ \ \forall i = 1, \ldots, m \right\}. \quad (4.10)$$

The first two conditions imply the feasibility of the resource allocation, and the third condition restricts the set to the extreme points of the unit hypercube around $\omega^t$. It is important to notice here that within the described unit hypercube around $\omega^t$, the functions $\tilde{z}_i^t$ behave as linear functions, and hence $\mathbf{E}\left[K_i^t\right] = \omega^t$ implies $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$. Clearly, $\omega^t$ lies within the convex hull of the points in $\mathcal{K}^{\omega^t}$ and it can be described as a convex combination of the points in $\mathcal{K}^{\omega^t}$. However, the number of elements in $\mathcal{K}^{\omega^t}$ is not necessarily polynomial. The Caratheodory theorem states that there exists a subset $\mathcal{K}^t$ of $\mathcal{K}^{\omega^t}$ such that $|\mathcal{K}^t| = m$ and $\omega^t$ can be described as a convex combination of the points in $\mathcal{K}^t$. The resource allocation $K^t$ is selected randomly from the set $\mathcal{K}^t$ with probabilities

equal to the weights in this convex combination. The subroutine described in Figure 4.4 simultaneously constructs the set $\mathcal{K}^t$ and randomly selects one element in the set with the probability of each element being the respective weight in the convex combination.

1: Set $K^t = \omega^t$, let $f_i$ denote $K_i^t - \lfloor K_i^t \rfloor$
2: $\delta = \sum_{i=1}^{m} f_i$
3: **if** $\delta = 0$ **then**
4: **return** $K_i^t$
5: **end if**
6: $I^+ =$ indices of the first $\delta$ fractional components of $K^t$ ,
$I^- =$ indices of the remaining fractional components of $K^t$
7: $i^+ = \arg\min_{i \in I^+} \{f_i\}$
$i^- = \arg\min_{i \in I^-} \{1 - f_i\}$
8: $\theta = \min\{f_{i^+}, 1 - f_{i^-}\}$,
$i^* = i^+$ if $\theta = f_{i^+}$, $i^* = i^-$ if $\theta = 1 - f_{i^-}$
9: **if** $i = i^*$ **then**
10: w.p. $\theta$: ***Share***("Terminate"), w.p. $1 - \theta$: ***Share***("Continue")
11: **else**
12: *Status* = ***Receive( )***
13: **end if**
14: **if** Status = Terminate **then**
15: $K_i^t = \lceil K_i^t \rceil \ \forall i \in I^+, K_i^t = \lfloor K_i^t \rfloor \ \forall i \in I^-$
16: **else if** Status = Continue **then**
17: $K_i^t = \frac{K_i^t - \theta \lceil K_i^t \rceil}{1-\theta} \ \forall i \in I^+, K_i^t = \frac{K_i^t - \theta \lfloor K_i^t \rfloor}{1-\theta} \ \forall i \in I^-$
18: **end if**
19: **go to 2**

Figure 4.4: The ***Pick*** $K_i^t$ ***Randomly*** subroutine

We illustrate the steps of the ***Pick*** $K_i^t$ ***Randomly*** subroutine with the following example: Let $K^t = f = [0.3, 0.8, 0.9]$, and therefore $\delta = 0.3 + 0.8 + 0.9 = 2$. The set of indices of the first $\delta$ fractional components is $I^+ = \{1, 2\}$ and the set of remaining fractional indices is $I^- = \{3\}$. We round up $f_1$ and $f_2$ and round down $f_3$ to get the integer vector $[1, 1, 0]$. The goal is find the vector $y$ such that $[0.3, 0.8, 0.9] = \theta[1, 1, 0] + (1 - \theta)y$. The constraint $y_i \geq 0$ is restrictive for $i \in I^+$, i.e. $\theta \leq f_i$, and we get $i^+ = 1$ as the player with the tightest bound. The constraint $y_i \leq 1$ is restrictive for $i \in I^-$, i.e. $\theta \leq 1 - f_i$, and we get $i^- = 3$ as the player with the tightest bound. The minimizer $\theta = 0.1$ and the player

with the tightest bound $i^*$ is player 3. At this point, player 3 randomly selects to either terminate with probability $\theta = 0.1$, or to continue with probability $1 - \theta = 0.9$. If the subroutine terminates, then $K^t$ is the integer point $[1, 1, 0]$. Otherwise, the subroutine is repeated for $K^t = y = [2/9, 7/9, 1]$. Notice that if player 3 chooses to continue, in the remaining iterations $K_3^t$ is fixed to 1, and therefore player 3 will no longer be a part of the random selection process and $K_3^t = 1$.

**Theorem 4.3.** *The* **Pick**$K_i^t$**Randomly** *subroutine described in Figure 4.4 returns a point* $K^t$ *such that* $\mathbf{E}[K^t] = \omega^t$ *and guarantees* $\mathbf{E}[z_i^t(K_i^t)] = \tilde{z}_i^t(\mathbf{E}[K_i^t])$.

*Proof.* The underlying idea in Figure 4.4 is along the lines of the proof of the Caratheodory theorem. Given a non-extreme point $y^0$, we want to find integer points $\bar{K}^1, \ldots, \bar{K}^m \in \mathcal{K}^{\omega^t}$, and weights $\lambda_1, \ldots, \lambda_m \in [0, 1]$ that satisfy $y^0 = \sum_{j=1}^m \lambda_j \bar{K}^j$ and $\sum_{j=1}^m \lambda_j = 1$, i.e. $y^0 \in conv(\{\bar{K}^j\}_{j=1}^m)$. Let us start with the induction $y^0 = \theta^1 \bar{K}^1 + (1 - \theta^1) y^1$, where $\bar{K}^1 \in \mathcal{K}^{\omega^t}$ and $y^1 \in conv(\mathcal{K}^{\omega^t})$. In other words, let us fix one of the integer points, $\bar{K}^1$, find the possibly fractional point $y^1$ on the other end of the line segment passing through $y^0$ and $\bar{K}^1$. $\mathcal{K}^{\omega^t}$ is an $m - 1$ dimensional polytope, and since $y^0 \in conv(\mathcal{K}^{\omega^t})$, it can be expressed as a convex combination of $m$ points, i.e. $y^0 \in conv(\{\bar{K}^j\}_{j=1}^m)$. In order for this recursion to make sense, $y^1$ should be describable by $m - 1$ points in $\mathcal{K}^{\omega^t}$. Therefore our goal, given $y^0$ and $\bar{K}^1$, is to find the maximum value $\theta^1$ can get such that the components of $y^1$, $y_i^1 = \frac{y_i^0 - \theta^1 \bar{K}_i^1}{1 - \theta^1}$, are within feasible bounds, in our case $[\lfloor y_i^0 \rfloor, \lceil y_i^0 \rceil]$. Noticed that when $\theta^1$ is set to the maximum feasible value, at least one component of $y^1$ is fixed to one of the bounds, and hence we get the reduction in dimension. Lines 7 and 8 compute this maximum feasible value for $\theta^1$.

The algorithm starts with $y^0 = \omega^t$ and uses the recursion $y^{r-1} = \theta^r \bar{K}^r + (1 - \theta^r) y^r$, where $y^r \in conv(\{\bar{K}^j\}_{j=r+1}^m)$. With this recursive method, we get $y^0 = \sum_{j=1}^m \lambda_j \bar{K}^j$, where the probability of selecting $\bar{K}^j$ is $\lambda_j = \theta^j \prod_{r=1}^{j-1}(1 - \theta^r)$ for $j = 1, \ldots, m - 1$ and $\lambda_m = \prod_{r=1}^{m-1}(1 - \theta^r)$. At each iteration $\bar{K}^r$ is defined such that the first $\delta$ fractional indices of $y^{r-1}$ are rounded up and the remaining are rounded down, where $\delta$ is described

as in Line 2. The ***Pick***$K_i^t$***Randomly*** subroutine does not necessarily compute all candidates $\bar{K}^1, \ldots, \bar{K}^m$, it concurrently carries out the process of construction of those points and the random selection. For $r = 1, \ldots, m-1$ it selects a point $\bar{K}^r$ right after its construction with probability $\lambda^r$. Namely, with probability $\prod_{j=1}^{r-1}(1-\theta^j)$ the previous $r-1$ points have not been selected, i.e. Status = Continue, and after that with probability $\theta^r$ $\bar{K}^r$ is selected, i.e. Status = Terminate. Hence we get $\lambda_r = \theta^r \prod_{j=1}^{r-1}(1-\theta^j)$. If none of the first $m-1$ points are selected, i.e. with probability $\lambda_m = \prod_{r=1}^{m-1}(1-\theta^r)$, then $\bar{K}^m$ is selected. This means that the expected output of the ***Pick***$K_i^t$***Randomly*** subroutine is $\omega^t$.

Furthermore, notice that the points $\bar{K}^1, \ldots, \bar{K}^m \in \mathcal{K}^{\omega^t}$, meaning that the coordinates satisfy $\bar{K}_i^r \in [\lfloor \omega_i^t \rfloor, \lceil \omega_i^t \rceil]$ for all $r = 1, \ldots, m$. Within this unit hypercube, the piecewise linear objective functions $\tilde{z}_i^t$ act as linear functions. Due to this linearity, we have $\mathbf{E}\left[z_i^t(K_i^t)\right] = \tilde{z}_i^t\left(\mathbf{E}\left[K_i^t\right]\right)$.

□

It is also worth noting that the exact value of $\omega^t$ is not necessary for the ***Pick***$K_i^t$***Randomly*** subroutine. Using the fractional vector $f^t = \omega^t - \lfloor \omega^t \rfloor$ during the computations, i.e. $y^0 = f^t$, and finally shifting the resulting binary vector by $\lfloor \omega^t \rfloor$ yields the same result. This approach doesn't reduce the communication complexity, however it is advantageous in terms of privacy of data. The players only know the fractional components of other players' resource allocation rather than the resource allocation itself.

**Theorem 4.4.** *When* $\eta = \sqrt{\frac{2\ln m}{T}}$ *and* $z_i^t \in \mathcal{F}$ *such that* $c_{ij}^t \in [0,1]$*, the expected regret of the decentralized algorithm described in Figure 4.3 is upper bounded by* $O(K\sqrt{T \ln m})$.

*Proof.* Let $K^*$ be the best static decision described in (4.4), and $\Delta(x||y) = \sum_{i=1}^m x_i \ln \frac{x_i}{y_i} + y_i - x_i$ be the relative entropy function. Recall that the upper envelope of the functions $z_i^t \in \mathcal{F}$, i.e. $\tilde{z}_i^t$, are piecewise linear and concave, and therefore can be expressed as the minimum of $K$ linear functions. $\tilde{z}_i^t(K_i) = \min_{\{j=1,\ldots K\}} \left\{\beta_{ij}^t + c_{ij}^t K_i\right\}$, where $\beta_{ij}^t = \sum_{k=1}^j (c_{ik}^t - c_{ij}^t)$. Let $\beta_i^{*t} + c_i^{*t} K_i^*$ denote the line that is the minimizer for $K_i = \omega_i^{t-1}$. Recall once again that

$z_i^t(K_i) = \tilde{z}_i^t(K_i)$ for integer $K_i$ values. Consider the chain of inequalities:

$$\Delta(K^*\|\omega^t) - \Delta(K^*\|\omega^{t-1}) + \eta\left(\sum_{i=1}^m \min_{\{j=1,\ldots K\}}\left\{\beta_{ij}^t + c_{ij}^t K_i^*\right\}\right) \tag{4.11}$$

$$\leq \Delta(K^*\|\hat{\omega}^t) - \Delta(K^*\|\omega^{t-1}) + \eta\left(\sum_{i=1}^m \beta_i^{*t} + c_i^{*t} K_i^*\right) \tag{4.12}$$

$$= \sum_{i=1}^m \omega_i^{t-1}(e^{\eta c_i^{*t}} - 1) + \eta\beta_i^{*t} \tag{4.13}$$

$$\leq \sum_{i=1}^m \omega_i^{t-1} c_i^{*t}(e^{\eta} - 1) + \eta\beta_i^{*t} \tag{4.14}$$

$$\leq (e^{\eta} - 1)\left(\sum_{i=1}^m c_i^{*t}\omega_i^{t-1} + \beta_i^{*t}\right). \tag{4.15}$$

For iteration $t$, (4.11) denotes the difference of the relative entropies of the best static decision $K^*$ with respect to $\omega^t$ and $\omega^{t-1}$, plus $\eta$ times the value of the best static decision at iteration $t$, i.e. $\sum_{i=1}^m z_i^t(K_i^*)$. Among the linear functions, $\beta_i^{*t} + c_i^{*t}K_i^*$ is an upperbound on the minimizer and we get (4.12). Expanding the $\Delta$ functions gives (4.13). Due to our assumption $c_i^{*t} \in [0, 1]$, (4.14) is an upper bound on (4.13). We can further bound this term by (4.15). Notice that the term in the summation in (4.15) is equal to $\tilde{z}_i^t(\mathbf{E}[K_i^t])$, since the line $\beta_i^{*t} + c_i^{*t}K_i^*$ is defined to be the minimizer at $K_i = \omega_i^{t-1}$. By the property of our algorithm, this is equal to $\mathbf{E}[z_i^t(K_i^t)]$. Hence, for iteration $t$, we have

$$\Delta(K^*\|\omega^t) - \Delta(K^*\|\omega^{t-1}) + \eta\sum_{i=1}^m z_i^t(K_i^*) \leq (e^{\eta} - 1)\sum_{i=1}^m \mathbf{E}[z_i^t(K_i^t)]. \tag{4.16}$$

Summing (4.16) over $t = 1, \ldots, T$ gives

$$\Delta(K^*\|\omega^T) - \Delta(K^*\|\omega^0) + \eta\sum_{t=1}^T\sum_{i=1}^m z_i^t(K_i^*) \leq (e^{\eta} - 1)\sum_{t=1}^T\sum_{i=1}^m \mathbf{E}[z_i^t(K_i^t)]. \tag{4.17}$$

By rearranging (4.17) and plugging in $\eta = \sqrt{2(\ln m)/T}$, we get the desired bound.

$$\begin{aligned} R_{\mathcal{A}}^{E}(T) &\leq \frac{\Delta(K^*\|\omega^0)}{e^{\eta}-1} + \left(1 - \frac{\eta}{e^{\eta}-1}\right) \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) \\ &\leq K\sqrt{2T\ln m} \end{aligned}$$

□

### 4.6 Experimental Results

The $K\sqrt{2T\ln m}$ upper bound on the expected regret is theoretically shown to be correct, however a natural question to ask is its tightness on average. In order to answer this question, we conduct experiments using the settings explained in the following subsections. These experiments also provide additional insight about how the expected regret function behaves throughout time within the known upper bound. Furthermore, we provide the realized regret values alongside the expected regret values to observe discrepancies both in values and in function behaviors.

For every experiment, we report three measures:

1. $\omega^t$ vector: How the expectation vector, $\mathbf{E}\left[(K^t)\right]$ changes over time.

2. Expected Regret: Using the $K^*$ definition in (4.4), for every time period $t = 1, \ldots, T$, we report the following as the expected regret:

$$\sum_{t'=1}^{t} \sum_{i=1}^{m} z_i^{t'}(K_i^*) - \sum_{t'=1}^{t} \sum_{i=1}^{m} \mathbf{E}\left[z_i^{t'}(K_i^{t'})\right].$$

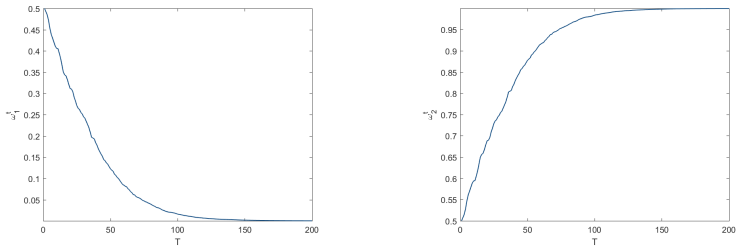
3. Realized Regret: Using the $K^*$ definition in (4.4), for every time period $t = 1, \ldots, T$,

we report the following as the realized regret:

$$\sum_{t'=1}^{t}\sum_{i=1}^{m} z_i^{t'}(K_i^*) - \sum_{t'=1}^{t}\sum_{i=1}^{m} z_i^{t'}(K_i^{t'}).$$

### 4.6.1 Experimental Setting 1

We consider a setting with two players, $m = 2$, and a single unit of resource $K = 1$. The objective functions are generated such that player 1 always gets the invaluable item, $c_1^t \sim U(0, 0.5)$, and player 2 gets the valuable item, $c_2^t \sim U(0.5, 1)$. We observe this online game for $T = 200$ iterations. Clearly, the best static decision $K^*$ is $(0, 1)$.



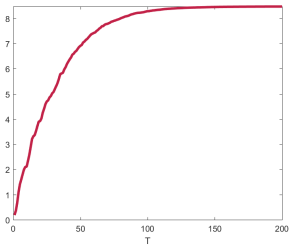

(a) Player 1 (b) Player 2

Figure 4.5: $\omega^t$ versus time

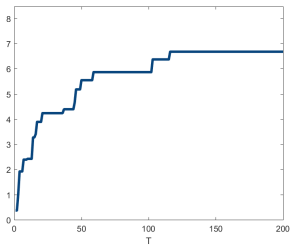
As shown in Figure 4.5, the expectation vector $\omega^t$ steadily converges to $K^*$. In addition to this convergence, we also observe a $\sqrt{T}$ shape in the expected regret over time, in Figure 4.6a. Notice that the theoretical upper bound on the expected regret $K\sqrt{2T \ln m}$ is 11.77, whereas the expected regret converges to 8.67. As $\omega^t$ converges to $K^*$, the probability of $K^t = K^*$ increases. We can observe this in Figure 4.6b, where for long periods of time the expected regret stays constant when $K^t = K^*$, but every now and then when the algorithm chooses the other solution due to randomness a jump in the realized regret value occurs.

### 4.6.2 Experimental Setting 2

In this setting, we again consider a setting with two players, $m = 2$, and a single unit of resource $K = 1$. We construct the second experimental setting such that the two static



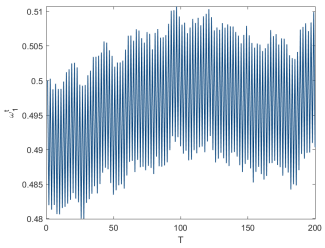

(a) Expected Regret



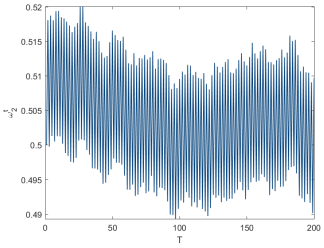

(b) Realized Regret

Figure 4.6: Expected and realized regret versus time

decisions $(1, 0)$ and $(0, 1)$ do not have a significant dominance over each other. Namely, for $t$ odd, player 1 gets the valuable item, $c_1^t \sim U(0.9, 1)$, and player 2 gets the invaluable item, $c_2^t \sim U(0, 0.1)$, and for $t$ even, player 2 gets the valuable item, $c_2^t \sim U(0.9, 1)$, and player 1 gets the invaluable item, $c_1^t \sim U(0, 0.1)$. We observe this online game for $T = 200$ iterations.




(a) Player 1

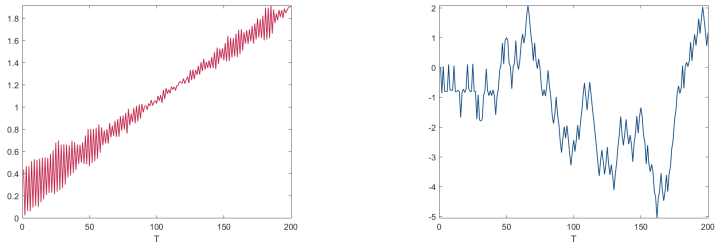



(b) Player 2

Figure 4.7: $\omega^t$ versus time

As shown in Figure 4.7, the expectation vector $\omega^t$ oscillates around $(0.5, 0.5)$. This oscillation causes a linear trend in the expected regret, which can be observed in Figure 4.8a. A natural question to ask as a follow up is whether this linear increase over time will eventually exceed the $O(\sqrt{T})$ bound as $T$ increases. The answer is that it will not, by the following argument: The slope of the expectation, which is a function of the length of the oscillation interval around $(0.5, 0.5)$ depends on the step-length parameter $\eta$. Recall that the definition $\eta = \sqrt{\frac{2 \ln m}{T}}$ includes the termination time for the online algorithm. Hence, if we mean to carry on for a longer period of time $T' > T$, $\eta$ will be a smaller value, hence the length of the oscillation interval and the slope of the linear trend in expected regret

will decrease accordingly, keeping the final value of the expected regret below the now $O(\sqrt{T'})$ threshold. Notice that the upper bound on the expected regret is the same as the experimental setting 1, since $m$, $K$, and $T$ values are constant; however for this setting it is a less tight bound.




(a) Expected Regret (b) Realized Regret

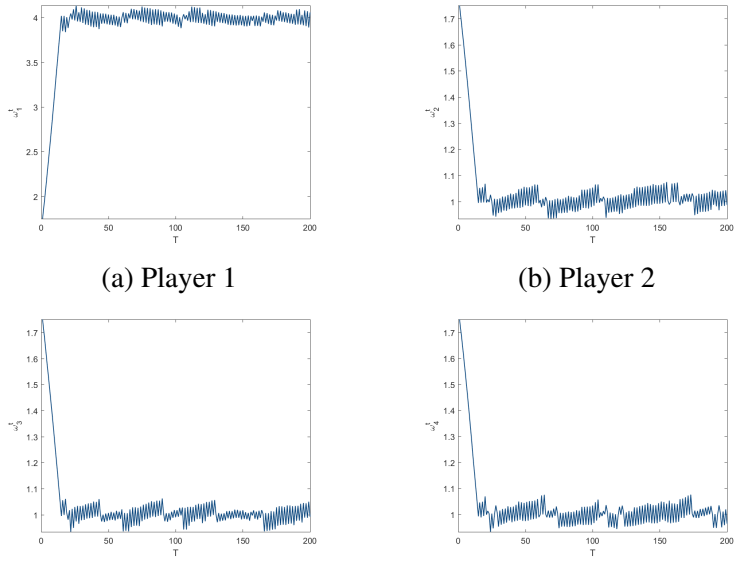
Figure 4.8: Expected and realized regret versus time

We observe an interesting phenomenon in Figure 4.8b when we look at the realized regret. Recall that the player with the valuable item alternates at every iteration. So of the two possible candidates for the best static decision, each of them receives the valuable item at exactly $T/2$ iterations. The expectation vector $\omega^t$ assigns almost equal probabilities to each player. This means that in reality, due to this randomness, there may be periods of time when the random resource allocation indeed matches the assignment of the valuable item, hence being more successful than the best static decision. This appears in Figure 4.8b as regions when the regret value decreases, and sometimes even falls below 0.

### 4.6.3 Experimental Setting 3

So far we have observed a setting where best static decision dominates other resource allocations, in which case the expectation vector $\omega^t$ converges to the best static decision, and a setting where the two decisions are not significantly different, in which case the expectation vector $\omega^t$ oscillates around the center. We construct the third setting such that there exists a best static decision that dominates other resource allocations, but which is not an extreme point of the feasible resource allocations polytope. We consider a setting with

four players, $m = 4$, and seven units of resource $K = 7$. The value of the items are such that at every iteration, players 1, 2, 3, and 4 receive 4, 1, 1, and 1 valuable items respectively, i.e. $c_{ij}^t \sim U(0.9, 1)$. For the remaining items we have $c_{ij}^t \sim U(0, 0.1)$. The number of valuable items equals the total amount of resournce, hence we have $K^* = (4, 1, 1, 1)$. We observe this online game for $T = 200$ iterations.



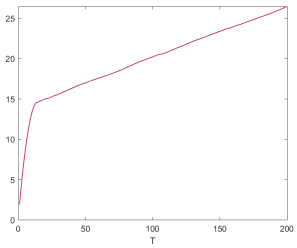
(a) Player 1 (b) Player 2

(c) Player 3 (d) Player 4

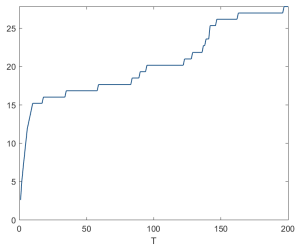
Figure 4.9: $\omega^t$ versus time

Figure 4.9 shows that once $\omega^t$ reaches a certain neighborhood of $K^*$, it starts oscillating around it, and the length of that neighborhood depends on $T$. More specifically, the algorithm reaches $\omega^t = (4 - \epsilon_1, 1 + \epsilon_2, 1 + \epsilon_3, 1 + \epsilon_4)$, and the objective coefficients force $\omega_1^t$ to increase and the remaining to decrease. Hence at iteration $t + 1$ we get $\omega^{t+1} = (4 + \epsilon‘_1, 1 - \epsilon‘_2, 1 - \epsilon‘_3, 1 - \epsilon‘_4)$. Now the objective coefficients force $\omega_1^{t+1}$ to decrease and the remaining to increase, pushing $\omega_1^{t+2}$ slightly below 4, and the remaining slightly above 1. Because it is impossible to get $\omega^t = (4, 1, 1, 1)$, this oscillation continues.

We again observe the effect of the oscillation as a linear trend in the expected regret in Figure 4.10a. The previous argument about the linear trend exceeding the $\sqrt{T}$ upper bound still holds. With this parameter set, the value of the upper bound $K\sqrt{2T \ln m}$ is 116.56,




(a) Expected Regret




(b) Realized Regret

Figure 4.10: Expected and realized regret versus time

which is a very loose bound for this parameter set. In the realized regret shown in Figure 4.10b, we observe a similar trend to the realized regret in experimental setting 1. Namely, $\omega^t$ gets arbitrarily close to $K^*$, meaning with high probability the algorithm will output $K^t = K^*$, in which case the realized regret value stays constant, and occasionally when $K^t$ deviates from $K^*$ due to randomness, we observe a sharp increase in the realized regret value.

## 4.7 Generalization of the Problem Structure

So far, we analyzed the performance of the online optimization algorithms with respect to the cardinality problem (4.2). However, recall that in Section 2.4 where we defined the *concavity property*, we showed that any problem that satisfies the concavity property can be expressed as a cardinality problem. Therefore, the results in this chapter can be generalized to any problem that satisfies the concavity property, i.e. any problem whose objective function is in $\mathcal{F}$, the examples of which are provided in Section 2.4.1.

## 4.8 Using Approximations

$\alpha$-regret is a notion of performance used in settings involving approximation functions [37]. In this line of thought, we define expected $\alpha$-regret in the following way:

$$R_{\mathcal{A}}^{\alpha E}(T) = \sup_{z_i^t \in \mathcal{F}} \left\{ \alpha \sum_{t=1}^{T} \sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T} \sum_{i=1}^{m} \mathbf{E}\left[ z_i^t(K_i^t) \right] \right\}. \tag{4.18}$$

Assume we have a function $\bar{z}_i^t \in \mathcal{F}$ that satisfies the following with respect to the true objective function $z_i^t$:

$$\rho_i^L \bar{z}_i^t(K_i) \leq z_i^t(K_i) \leq \rho_i^U \bar{z}_i^t(K_i). \tag{4.19}$$

Let $\rho_{min}^L = \min_i \rho_i^L$ and $\rho_{max}^U = \max_i \rho_i^U$. For any online convex optimization algorithm, the upper bound on expected regret with respect to $\bar{z}_i^t$ that holds by Theorem 4.2 is a valid upper bound on expected $\alpha$-regret with respect to $z_i^t$, where $\alpha = \frac{\rho_{min}^L}{\rho_{max}^U}$. This holds by the following argument:

$$\begin{aligned}
o(T) &\geq \sum_{t=1}^{T}\sum_{i=1}^{m} \bar{z}_i^t(\bar{K}_i^*) - \sum_{t=1}^{T}\sum_{i=1}^{m} \mathbf{E}\left[\bar{z}_i^t(K_i^t)\right] && (4.20)\\
&\geq \sum_{t=1}^{T}\sum_{i=1}^{m} \bar{z}_i^t(K_i^*) - \sum_{t=1}^{T}\sum_{i=1}^{m} \mathbf{E}\left[\bar{z}_i^t(K_i^t)\right] && (4.21)\\
&\geq \sum_{t=1}^{T}\sum_{i=1}^{m} \frac{z_i^t(K_i^*)}{\rho_i^U} - \sum_{t=1}^{T}\sum_{i=1}^{m} \mathbf{E}\left[\frac{z_i^t(K_i^t)}{\rho_i^L}\right] && (4.22)\\
&\geq \frac{1}{\rho_{max}^U}\sum_{t=1}^{T}\sum_{i=1}^{m} z_i^t(K_i^*) - \frac{1}{\rho_{min}^L}\sum_{t=1}^{T}\sum_{i=1}^{m} \mathbf{E}\left[z_i^t(K_i^t)\right] && (4.23)\\
&= \frac{1}{\rho_{min}^L}\left(\frac{\rho_{min}^L}{\rho_{max}^U}\sum_{t=1}^{T}\sum_{i=1}^{m} z_i^t(K_i^*) - \sum_{t=1}^{T}\sum_{i=1}^{m} \mathbf{E}\left[z_i^t(K_i^t)\right]\right). && (4.24)
\end{aligned}$$

Let $K^*$ be the best static decision defined in (4.4), and $\bar{K}^*$ be defined as

$$\bar{K}^* = \arg\max\left\{\sum_{t=1}^{T}\sum_{i=1}^{m} \bar{z}_i^t(K_i) \;:\; \sum_{i=1}^{m} K_i = K,\;\; K_i \geq 0\;\forall i = 1,\ldots,m\right\}. \tag{4.25}$$

(4.20) states that since $\bar{z}_i^t \in \mathcal{F}$, the expected regret of the selected online convex optimization algorithm is upper bounded by a sublinear function of $T$, by Theorem 4.2. We replace the maximizer $\bar{K}^*$ with $K^*$ to get (4.21). Using the definition in (4.19), we lower

bound this term by (4.22). By replacing the $\rho_i^U$ with $\rho_{max}^U$ and $\rho_i^L$ with $\rho_{min}^L$ we get (4.23) and rearranging the terms gives us the desired expected $\alpha$-regret where $\alpha = \frac{\rho_{min}^L}{\rho_{max}^U}$.

This result can be utilized in the following way: assume that the true objective function value for our problem at every iteration is difficult to compute, however there exists a concave approximation algorithm to solve our problem. In such cases, by using an online convex optimization algorithm on the approximation function, we can get an upper bound of $o(T)$ on the expected $\alpha$-regret. Examples of such concave approximations functions are discussed in Section 3.2.3.

## 4.9 Conclusions

In this chapter, we focused on the problem of resource allocation in a decentralized online setting. We proved that there exists no deterministic online algorithms that will have sublinear regret bounds when applied to our problem. We further investigated randomized algorithms, and reduced our problem to an online convex optimization problem by using linear relaxations. With this observation, we derived a decentralized randomized algorithm that has an expected regret bound of $O(\sqrt{T})$. This algorithm can be generalized to any problem that satisfies our concavity property.

# CHAPTER 5
# CONCLUSIONS

In this thesis we focus on distributed integer programming problems that are separable over the players except for a single knapsack type constraint. The goal is designing decentralized algorithms for allocating the resource to players with minimal communication among them. Our algorithms are designed for settings with cardinality constraint as the binding coupling constraint, or with arbitrary knapsack coupling and a cardinality objective. The algorithms have optimality guarantees for problems that satisfy the concavity property. We provide sufficient conditions for problems to guarantee optimality of the algorithms, and give examples of problems that satisfy these conditions. These algorithms terminate in linear time.

For algorithms that do not have the concavity property, we propose using approximation algorithms. We provide upper bounds on the error of the decentralized algorithms when approximation functions are used instead of true objective function values. Furthermore, our decentralized algorithms can be modified to adapt to certain greedy algorithms. In the absence of the concavity property, the performance of our algorithms can be arbitrarily bad. However, experiments show that for some initial point selection methods, the average error of our algorithms are within 1 %.

Finally, we consider the same problem in an online setting. We show that there exists no deterministic online algorithms that will have sublinear regret bounds when applied to our problem. As for randomized algorithms, we reduce our problem to an online convex optimization problem, and propose a randomized algorithm along these lines that matches the known state of the art regret bound for online convex optimization algorithms. This algorithm can be generalized to any problem with a concave structure.

The main property exploited by all the algorithms discussed in this thesis is the con-

cavity property. We focus our future research on analyzing the concavity property more thoroughly. One significant aspect of the concavity property is its connection to polyhedral structure. As we have stated before, it is not sufficient for the concavity property, for the matrix used in the polyhedral description of the convex hull of feasible points to be a non-negative matrix, and its necessity is still an open problem. Furthermore, problems that have the concavity property for a family of objective function coefficients can be investigated further.

Another important area to focus along these lines is the concave approximation algorithms. So far we have shown that linear relaxation and the greedy algorithm are examples of concave approximation algorithms for several problems. These examples can be enriched including algorithms such as the modified greedy algorithm for the knapsack problem, which originally is not a concave approximation algorithm but our decentralized heuristics can be adapted to preserve the error bound accordingly.

We have shown that in the online setting, problems with the concavity property can be solved using various online convex optimization techniques in a randomized manner. One of our future goals is to devise randomized algorithms to solve problems without the concavity property, and observe how the regret bounds change accordingly.

## REFERENCES


[1] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004.

[2] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[3] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in mpc and network flows," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2015.

[4] S. Sundhar Ram, A. Nedic, and V. V. Veeravalli, "A new class of distributed optimization algorithms: Application to regression of distributed data," *Optimization Methods and Software*, vol. 27, no. 1, pp. 71–88, 2012.

[5] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[6] I. Lobel, A. Ozdaglar, and D. Feijer, "Distributed multi-agent optimization with state-dependent communication," *Mathematical Programming*, vol. 129, no. 2, pp. 255–284, 2011.

[7] R. L. Raffard, C. J. Tomlin, and S. P. Boyd, "Distributed optimization for cooperative agents: application to formation flight," *43rd IEEE Conference on Decision and Control*, vol. 3, pp. 2453–2459, 2004.

[8] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," *51st IEEE Conference on Decision and Control*, pp. 5445 –5450, 2012.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[10] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[11] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A distributed newton method for network optimization," *48th IEEE Conference on Decision and Control*, pp. 2736 – 2741, 2009.

[12] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[13] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: identifying representative elements in massive data," *Advances in Neural Information Processing Systems 26*, 2013.

[14] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward, "The power of randomization: Distributed submodular maximization on massive datasets," *CoRR*, vol. abs/1502.02606, 2015.

[15] G. Singh and C. M. O'Keefe, "Decentralised scheduling with confidentiality protection," *Operations Research Letters*, vol. 44, no. 4, pp. 514–519, 2016.

[16] E. Hazan, *Introduction to online convex optimization*, ser. Foundations and Trends(r) in Optimization Series. Now Publishers, 2016, ISBN: 9781680831702.

[17] S. Bubeck, "Introduction to online optimization," *Lecture Notes*, pp. 1–86, 2011.

[18] W. M. Koolen, M. K. Warmuth, J. Kivinen, *et al.*, "Hedging structured concepts," 2010.

[19] J.-Y. Audibert, S. Bubeck, and G. Lugosi, "Regret in online combinatorial optimization," *Mathematics of Operations Research*, vol. 39, no. 1, pp. 31–45, 2013.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions - i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[21] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, no. 1, pp. 7–66, 1992.

[22] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Athena Scientific, 1997.

[23] M. Rodeh, "Finding the median distributively," *Journal of Computer and System Sciences*, vol. 24, pp. 162–166, 1982.

[24] E. H. Aghezzaf and L. A. Wolsey, "Lot-sizing polyhedra with a cardinality constraint," *Operations Research Letters*, vol. 11, no. 1, pp. 13–18, 1992.

[25] G. L. Nemhauser and L. A. Wolsey, *Wiley Series in Discrete Mathematics and Optimization : Integer and Combinatorial Optimization (1)*. Somerset, US: Wiley-Interscience, 1999.

[26] A. Schrijver, *Combinatorial Optimization*. Berlin: Springer-Verlag, 2003, vol. A, B.

[27] M. Walter, P. Damcı-Kurt, S. S. Dey, and S. Küçükyavuz, "On a cardinality-constrained transportation problem with market choice," *Operations Research Letters*, vol. 44, no. 2, pp. 170–173, 2016.

[28] B. Korte and D. Hausmann, "An analysis of the greedy heuristic for independence systems.," *Algorithmic aspects of combinatorics*, vol. 2, pp. 65–74, 1978.

[29] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions - ii," *Polyhedral combinatorics*, pp. 73–87, 1978.

[30] M. Bläser, T. Heynen, and B. Manthey, "Adding cardinality constraints to integer programs with applications to maximum satisfiability," *Information Processing Letters*, vol. 105, no. 5, pp. 194–198, 2008.

[31] J. Lee, M. Sviridenko, and J. Vondrák, "Matroid matching: the power of local search," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 357–379, 2013.

[32] C. Chekuri, "The knapsack problem," *Lecture Notes*, 2009.

[33] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, pp. 5667–5681, 2010.

[34] G. Braun and S. Pokutta, "An efficient high-probability algorithm for linear bandits," *arXiv preprint arXiv:1610.02072*, 2016.

[35] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 28–34, 2002.

[36] A. Ben-Tal and A. Nemirovski, "Lectures on modern convex optimization," *Lecture Notes*, pp. 1–590, 2013.

[37] S. Kakade, A. T. Kalai, and K. Ligett, "Approximation algorithms going online," *Technical Report CMU-CS-07-102*, 2007.